N° d'enregistrement
au CNRS
A.O. 7420

THESE

présentée

A L'UNIVERSITÉ PARIS VI

par

Alexandre BRANDWAJN
Ingénieur ENST

pour obtenir

le GRADE de DOCTEUR-INGENIEUR

PROCEDURE D'EXPLOITATION POUR UNE LIAISON UC-UC

Soutenue le 4 octobre 1972, devant la Commission d'examen composée de :

MM. J. ARSAC — Président

R. DUPUY, P. FEAUTRIER — Examinateurs

l'enregistrement
au CNRS
A.O. 7420

THESE

présentée

A L'UNIVERSITÉ PARIS VI

par

Alexandre BRANDWAJN
Ingénieur ENST

pour obtenir

le GRADE de DOCTEUR-INGENIEUR

PROCEDURE D'EXPLOITATION POUR UNE LIAISON UC-UC

Soutenue le 4 octobre 1972, devant la Commission d'examen composée de :

MM. J. ARSAC — Président

R. DUPUY, P. FEAUTRIER — Examinateurs

UNIV.
LEIDEN
BIBL.

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41550025_0027

Je tiens à remercier :

Monsieur le Professeur ARSAC, qui m'a fait l'honneur d'accepter de présider le jury de thèse.

Monsieur le Professeur DUPUY, qui s'est chargé du rôle de Directeur de thèse.

Monsieur le Professeur FEAUTRIER,qui a bien voulu accepter de participer au jury.

Je dois une reconnaissance particulière à Mademoiselle FOUCAUD et à Madame LETAILLIEUR, qui se sont chargées de la réalisation de cet ouvrage.

Je remercie enfin l'IRIA pour m'avoir permis de réaliser l'application pratique sur l'ordinateur IRIS 50 du Centre de Calcul.

... et Iahvé dit : " Voici qu'eux tous forment un seul peuple et ont un seul langage. S'ils commencent à faire cela, rien désormais ne leur sera impossible de tout ce qu'ils décideront de faire."

L'Ancien Testament
Chapître XI

Procédure d'exploitation pour une liaison UC - UC

Alexandre BRANDWAJN

IRIA

1. - Introduction

2. - Définition de la procédure d'exploitation

2.1 - Commandes liées à l'établissement du sous-canal

2.11 - DEM1

2.12 - ADEM1

2.13 - NADEM1

2.14 - DEM2

2.15 - ADEM2

2.16 - NADEM2

2.2 - Commandes liées aux échanges en ligne

2.21 - ACC

2.22 - NACC

2.3 - Commandes de suspension des échanges

2.31 - RPL

2.32 - FERM

2.4 - Echanges en ligne-composition du message

2.41 - Zone de commandes

2.42 - Zone de données

3. - Etude générale du programme de téléconversion

3.1 - Fonctions assurées par le programme

3.2 - Organisations possibles du programme

3.21 - Fonction émission

3.22 - Fonction réception

3.2 - Structure hiérarchisée

3.31 - Interfaces standard

3.32 - Liaisons conceptuelles et niveaux

4. - Application à un système hypothétique

4.1 - Définition du système

4.2 - Remarques sur la synchronisation des processus à l'aide de sémaphores.

4.21 - Evénements simples

4.22 - Evénements composés

4.221 - Intersection des événements

4.222 - Réunion des événements

4.3 - Description du programme

4.31 - Racine

4.311 - Processus de sortie sur la machine à écrire

4.312 - Processus de réception et d'interprétation des messages MAE

4.313 - Etablissements des sous-canaux

4.32 - Fonction R

4.321 - Processus de réception et d'analyse des messages ligne

4.322 - Processus de sortie dans le fichier

4.33 - Fonction E

4.331 - Processus de garnissage du tampon composant

4.332 - File d'attente des sous-canaux SOURCE DE DONNEES prêts

4.333 - Processus de formation du message ligne

4.334 - Processus d'émission du message en ligne

5. - Extension de la procédure d'exploitation à un réseau de transmission

6. - Exemple d'application à l'ordinateur CII IRIS 50

6.1 - Objectifs

6.2 - Choix de l'ordinateur

6.21 - Généralités

6.22 - Système de gestion de fichiers

6.23 - Système de gestion de transmissions

6.3 - Régulateur des tâches

6.31 - Notion de tâche

6.32 - Fonction du régulateur

6.321 - Activation d'une tâche

6.322 - Lancement du Multitasking

6.323 - Introduction d'un mot dans la file d'attente d'une tâche

6.324 - Prélèvement d'un mot dans la file d'attente de la tâche

6.325 - Attente d'un événement parmi n

6.326 - Changement de la priorité d'une tâche

6.327 - Désactivation d'une tâche

6.328 - Remarques sur le régulateur

6.4 - Description du programme réalisé

6.41 - Généralités

6.42 - Entrée des messages sur la machine à écrire de pupitre

6.421 - Interruption pupitre

6.422 - Syntaxe du langage de commande

6.43 - Interprétation du langage de commande

6.44 - Lecture dans le fichier cartes

6.45 - Formation du message de transmission en ligne

6.46 - Emission en ligne

6.47 - Réception et interprétation du message de transmission

6.48 - Ecriture dans le fichier imprimante

6.49 - Remarques finales

annexes : 1 - attente d'un événement parmi n à l'aide de sémaphores

2 - régulateur des tâches

3 - langage de commande

1. Introduction

Dans ce travail nous voudrions proposer une procédure d'exploitation pour ce que les industriels appellent "une liaison UC - UC". Il s'agit d'une liaison directe entre les mémoires centrales de deux ordinateurs.

Par procédure d'exploitation nous entendons ici l'organisation logique des échanges d'information entre deux calculateurs en supposant qu'une procédure de transmission ait déjà été choisie pour les transferts d'information sur la ligne physique les reliant.*

Nous abordons ainsi un aspect particulier de l'important problème de la synchronisation des processus parallèles.

Pour des processus "cohabitant" dans le même espace-temps (mémoire-horloge) on commence à savoir assez bien le résoudre puisqu'on dispose alors d'un ensemble de mécanismes de synchronisation comme par exemple les sémaphores de Dijkstra [6].

On n'a pas encore trouvé de solution générale dans le cas, qui est le nôtre, de processus parallèles évoluant dans deux espace-temps différents et disjoints.

Les études que nous avons menées pendant quelques mois à la Compagnie Internationale pour l'Informatique avec Monsieur Jacques DEYCARD de l'Ecole Nationale Supérieure des Télécommunications constituent le point de départ de ce travail.

L'intérêt des réseaux d'ordinateurs, manifeste du point de vue conceptuel -ne serait-ce qu'en vertu d'une meilleure utilisation de la puissance des moyens de calcul-, ne l'est pas tout autant dès qu'on prend en considération l'aspect économique d'une telle installation puisqu'à l'heure actuelle le coût des lignes peut s'avérer prohibitif. Néanmoins, les réseaux intégrés de traitement de l'information semblent appelés à un développement important, surtout dans le cadre des grandes administrations comme le montrent les exemples des réalisations aux Etats-Unis : National Crime Information Center Network, Advanced Research Project Agency's Network, Public Health Service (projet), etc.

---

* Rappelons que l'on appelle procédure de transmission l'ensemble des conventions régissant la transmission de blocs d'information entre 2 correspondants sur une ligne de transmission de données.

Le problème peut être posé de la façon suivante : assurer des téléconversions de fichiers simultanément dans les deux sens tout en permettant aux opérateurs des deux calculatrices de communiquer entre eux par le truchement des machines à écrire de pupitre.*

L'une des difficultés majeures consiste à synchroniser les deux ordinateurs. En effet, il faut que lorsque l'un effectue une opération d'émission l'autre ordinateur soit prêt à la réception. La solution vient du fait de la simultanéité des traitements et des entrées/sorties. Chaque ordinateur sera en permanence en lecture ; il pourra ainsi recevoir à n'importe quel moment l'information que son correspondant voudra émettre.

On fixe une limite supérieure au nombre de conversions simultanées. Cette limite est indispensable si l'on veut être en mesure de garantir un temps de service acceptable.

L'utilisateur communiquera avec le programme par l'intermédiaire de la machine à écrire du pupitre.

Nous étudierons d'abord les aspects généraux de la procédure d'exploitation, nous poursuivrons ensuite de façon plus détaillée en nous plaçant dans le contexte d'un système hypothétique et nous essaierons, enfin, de construire une application simple à l'ordinateur IRIS 50 de la Compagnie Internationale pour l'Informatique fonctionnant sous le moniteur SIRIS 2.

Nous chercherons également à adapter la procédure d'exploitation à un réseau d'ordinateurs.

Précisons maintenant le sens de quelques termes que nous utiliserons tout au long de cet exposé :

- SOURCE DE DONNEES - ordinateur qui émet des données en ligne,
- PUITS DE DONNEES - ordinateur qui reçoit des données sur la ligne.

Dans une liaison il y a obligatoirement une SOURCE et un PUITS DE DONNEES. Un même ordinateur peut être puits pour une liaison et source pour une autre.

---

* il peut s'agir par exemple de la répartition de travail entre deux centres de calcul. Un programme déposé dans l'un d'entre eux sera exécuté dans l'autre.

- composants - correspond au mot anglais "device" (terminal ?),
- sous-canal - liaison entre deux composants rattachés à deux unités centrales différentes, plus généralement, entre deux fichiers.

Remarquons que la procédure d'exploitation ne remplace pas celle de transmission mais se situe à un niveau supérieur.

En effet, prenons comme exemple le standard TMM (Transmission Mode Message) de la Compagnie Internationale pour l'Informatique *.
Le mode "message" est un mode de communication où l'unité de données transmises est un ensemble limité de caractères : le bloc.

Un bloc d'informations est composé de caractères de procédure normalisés et spéciaux.
Aux transferts de blocs s'ajoutent des séquences de contrôle concernant la transmission.

Voici le format du bloc en procédure T2M option UC, code EBCDIC à 256 combinaisons (binaire) :

Ø SOH SEL NOB DLE STX texte binaire DLE $\begin{matrix}\text{ETB}\\\text{ETX}\end{matrix}$ CRC1 CRC2 PAD

Ø : représente la séquence de synchronisation nécessaire à la transmission de tout bloc.

SOH : indique le début d'un en-tête et du bloc le concernant
SEL : permet d'adresser la station du réseau invitée à recevoir
NOB : numérotation de blocs
DLE : sert à modifier la signification du caractère qui le suit
STX : sert à l'intérieur d'un bloc, à séparer l'en-tête du texte
ETB : marque la fin de l'un des blocs d'information du texte à transmettre
ETX : marque la fin du dernier bloc d'un texte
CRC1
CRC2 : 2 caractères de contrôle calculés sur le bloc d'information à partir d'un polynôme générateur.
PAD : est un caractère tampon.

La procédure de transmission se charge du bon déroulement des échanges sur la ligne.
Pour nous, dans la procédure d'exploitation, nous nous intéressons à la

---

* Nous nous inspirons de la note de programmation N° DSB/MS/71/034 de la CII

structure logique du "texte binaire" et à la gestion des liaisons (sous-canaux) tout en faisant appel à la procédure de transmission pour les E/S sur la ligne.

Dans le chapître suivant nous montrons comment, en partant de l'établissement d'un sous-canal, on arrive tout naturellement à introduire un certain nombre de commandes ou ordres échangés en ligne lesquels, liés à une structure donnée de l'information, permettent aux deux correspondants de se synchroniser et de bien se comprendre.

2. Définition de la procédure d'exploitation

2.1 Commandes liées à l'établissement du sous-canal

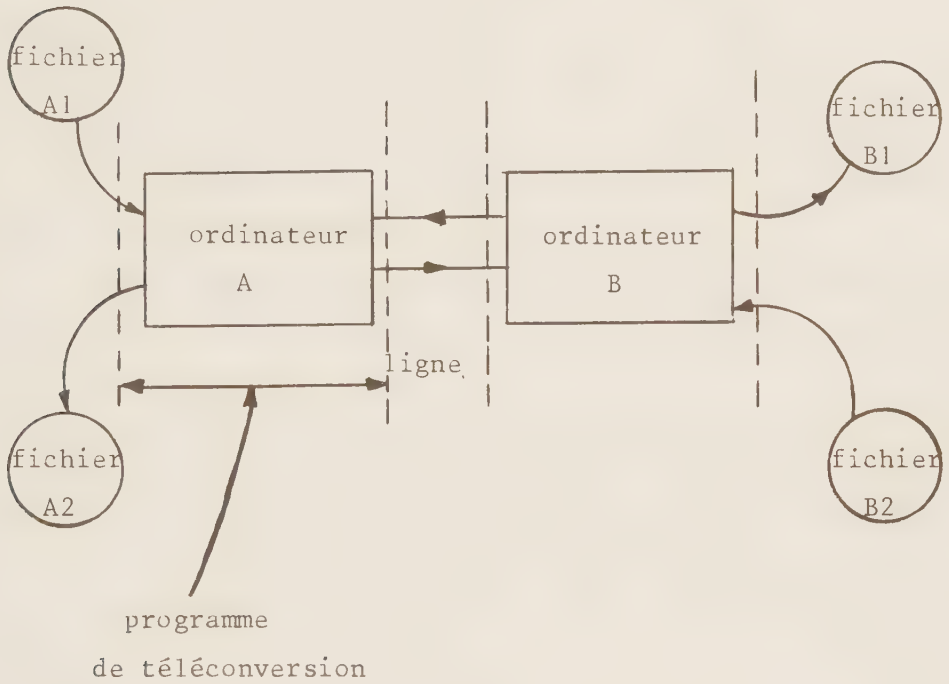


Figure 1 - Exemple de configuration

La figure 1 montre la place de notre programme qui se trouve entre 2 niveaux :
le niveau fichier et le niveau ligne de transmission reliant les deux ordinateurs.

Nous laisserons de côté les problèmes posés par l'accès aux fichiers et à la ligne celui-ci étant assuré, comme nous l'admettons d'avance, respectivement par un système de gestion de fichiers et par un système de gestion des transmissions.

Si une demande d'établissement du sous-canal est introduite à la MAE de l'un des 2 ordinateurs celui-ci cherchera à la satisfaire, ce qui implique qu'un certain nombre de conditions soient réunies :

1. on vérifie si le nombre maximum de sous-canaux est déjà atteint ; ce nombre est un paramètre du programme et il serait intéressant de le déterminer expérimentalement en fonction du temps de service.
   Avec la réponse positive s'offrent plusieurs solutions :
   a) on rejette la demande en avertissant le demandeur par un message à la console opérateur ; l'utilisateur pourra alors recommencer au bout d'un certain temps,
   b) on place d'office la demande dans une file d'attente et l'on cherchera à la satisfaire dès qu'un sous-canal deviendra disponible,
   c) on avertit le demandeur de l'impossibilité dans l'immédiat de satisfaire sa demande et on lui laisse le choix entre renoncer ou faire entrer sa demande dans une file d'attente.

   Au cas où le nombre de sous-canaux n'a pas atteint son maximum on vérifie si :
2. le terminal concerné est disponible
3. le fichier est disponible
4. on dispose de place mémoire suffisante pour créer les différents tampons nécessaires pour la gestion d'un sous-canal.
   Cette dernière condition peut se trouver implicitement satisfaite lorsque le nombre maximum de sous-canaux n'est pas dépassé, mais comme cela dépend de la façon dont on gère la mémoire dans le programme de téléconversion, il n'en est pas obligatoirement ainsi.

Si l'une de ces conditions venait à manquer, à cause d'une indisponibilité passagère, on pourrait, comme précédemment, laisser la décision au gré de l'utilisateur.

Il faut maintenant avertir l'ordinateur-correspondant qu'une nouvelle demande de liaison est arrivée ce qui lui permettra d'effectuer des opérations analogues à celles que nous venons de décrire.

Nous voyons apparaître ainsi un certain nombre de commandes (ordres) que les deux correspondants devront échanger pour bien se mettre d'accord, s'avertir mutuellement des conditions exceptionnelles de fonctionnement, etc...

Le premier type de commandes que nous considérerons concerne l'établissement du sous-canal.

Nous distinguerons deux demandes d'établissement suivant que le demandeur se trouve du côté de la future SOURCE ou du futur PUITS DE DONNEES, et nous les appellerons respectivement DEM1 et DEM2.

2.11 DEM1

L'ordre DEM1 est émis par le côté qui doit devenir, si la demande est satisfaite, SOURCE DE DONNEES.
DEM1 contient toutes les informations nécessaires à l'établissement du sous-canal comme le montre la figure 2 ci-dessous. Chaque case représente une "unité de codage" (nous admettrons que c'est un octet).

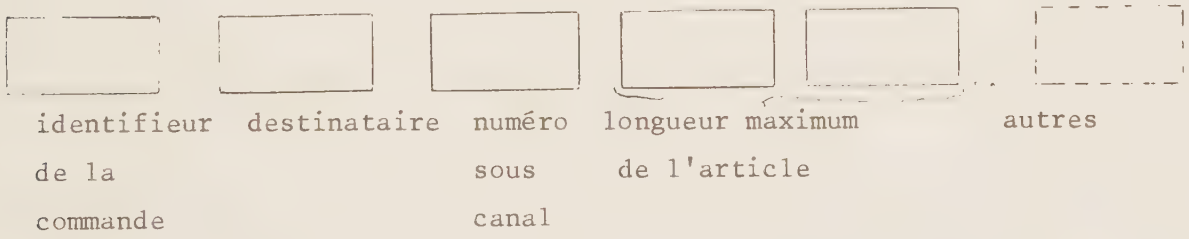


Figure 2. Composition de la commande DEM1.

La case destinataire détermine le terminal (device) qui servira de support au fichier à créer par le PUITS DE DONNEES.

La SOURCE potentielle assigne un numéro à la liaison, c'est par ce numéro que les deux côtés vont désormais identifier le sous-canal. On communique au futur PUITS DE DONNEES la longueur maximum d'article ainsi que d'autres informations comme par exemple le type de fichier à créer.

Dans certains cas des conversions de support seront indispensables ; nous conviendrons de les effectuer du côté du PUITS DE DONNEES.

La SOURCE devra, à cette fin, indiquer le numéro de conversion ; on suppose que chaque côté dispose de tables de conversion portant des numéros.

Signification de la commande : établir un sous-canal.

2.12 ADEM1

Examinons le cas de la demande DEM1 acceptée.

La figure 3 ci-dessous montre la composition de la réponse que nous appellerons ADEM 1 envoyée par le PUITS DE DONNEES.

identifieur de la commande | numéro du sous-canal | longueur allouée

Figure 3. Composition de la commande ADEM1.

La longueur allouée concerne le tampon à partir duquel va se faire l'écriture de l'article (ou des articles) dans le fichier ; cette longueur doit être un multiple de la longueur maximum d'article (voir algorithme d'allocation mémoire 4.313).

Signification de la commande : sous-canal établi.

2.13 NADEM1

La demande DEM1 rejetée entraîne l'envoi d'une réponse négative NADEM1. (voir figure 4).

identifieur de la commande | numéro sous-canal | motif

Figure 4. Composition de la commande NADEM1.

La case "motif" révèle les raisons du rejet de la demande DEM1 par le correspondant lesquels peuvent s'avérer de gravité inégale :

- manque place mémoire,
- indisponibilité du composant,
- panne du composant,
- composant inexistant, etc....

Comme pour une demande de l'opérateur il y a plusieurs attitudes à adopter :

a) rejeter la demande DEM1 en laissant à l'expéditeur le soin de recommencer,
b) former une file d'attente lorsque les motifs d'échec ne sont que passagers,
c) laisser le choix à l'opérateur.

Il est évident que la deuxième solution est plus souple et réduit éventuellement le trafic des demandes en ligne, mais elle complique notablement la gestion.

Signification de la commande : sous-canal non établi.

2.14 DEM2

L'ordre DEM2 est envoyé par le côté qui doit devenir, si la demande peut être satisfaite, PUITS DE DONNEES.

Remarquons tout de suite que l'introduction de cette commande pose quelques problèmes et l'on pourrait se contenter de la seule demande DEM1 grâce au sous-canal de communication entre les machines à écrire de pupitre des deux ordinateurs (en effet, il suffit de demander

à l'opérateur de faire envoyer DEM1 avec les paramètres convenables).

Regardons la figure 5 ci-dessous :

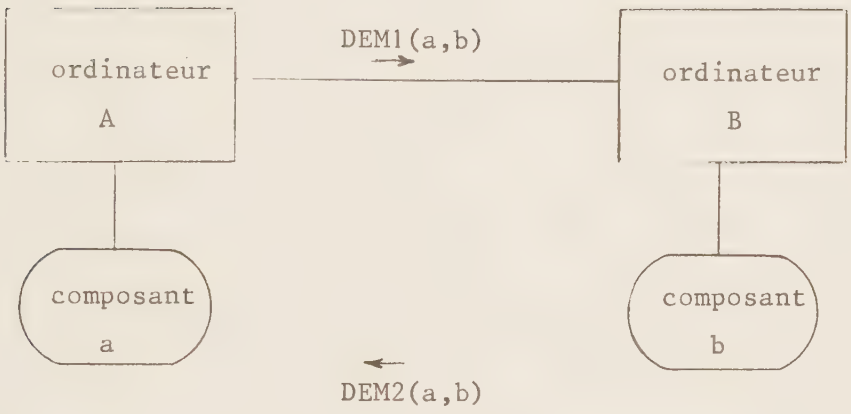


Figure 5.

La notation DEM(x,y) signifie : demande de sous-canal entre le composant x du côté de la SOURCE potentielle et le composant y du côté du PUITS.

Imaginons qu'en même temps l'ordinateur A envoie vers B l'ordre DEM1(a,b) et l'ordinateur B envoie vers A la commande DEM2(a,b).

Nous voyons bien qu'un conflit est possible et si l'on ne prend pas les précautions nécessaires on risque soit d'allouer deux fois le même terminal, soit de rejeter les deux demandes bien que l'une d'entre elle puisse être satisfaite. *

Une solution consiste à donner la priorité à l'une des demandes, par exemple à DEM1. On procéderait alors de la façon suivante :

Avant d'envoyer DEM1 on réserve le composant a du côté A; de l'autre côté, on crée une table des sous-canaux PUITS en cours d'établissement.

---

* Le même problème se présente si A envoie vers B DEM1(a,b) et B envoie vers A DEM2(a,c). En fait, il s'agit plutôt de l'attribution des fichiers aux sous-canaux que de celle des composants.

B assigne un numéro provisoire à la liaison dont il demande la création dans sa commande DEM2.

Si DEM1 parvient à B avant la réception d'une réponse positive c'est le sous-canal désigné par la commande DEM1 qui prend le pas.

Ceci conduit à prévoir pour la commande DEM2 la composition que montre la figure 6 ci-dessous :

Figure 6. Composition de la commande DEM2.

La case destinataire concerne le device qui sert de support au fichier dans lequel on veut lire à distance.

La longueur allouable précise la place mémoire dont on dispose pour créer un tampon composant.

D'autres informations qui dépendent du système de gestion de fichier devront également être transmises.

Signification de la commande : établir un sous-canal.

2.15 ADEM2

La future SOURCE DE DONNEES pour la liaison répondra par un accusé de réception positif ADEM2 si la demande DEM2 peut être acceptée,

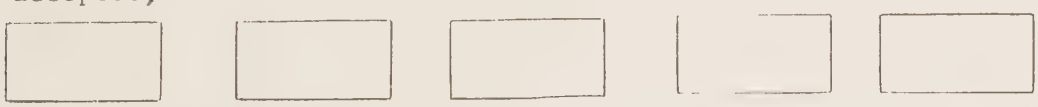


Figure 7. Composition de la commande ADEM2.

ce qui implique la satisfaction des conditions suivantes :

- composant existant et disponible
- fichier existant, disponible, normalement ouvert
- tampon composant créé
- longueur maximum allouable transmise dans DEM2 suffisante vu la longueur maximum d'article.

On voit sur la figure 7 ci-dessus que c'est la SOURCE DE DONNEES qui impose finalement le numéro du sous-canal ; c'est également elle qui indique la taille du tampon composant à retenir par le PUITS.

Signification de la commande : sous-canal établi.

2.16 NADEM2

La réponse de la SOURCE est NADEM2 si l'ordre DEM2 a dû être refusé.

identifieur de la commande | numéro sous-canal provisoire | motif

Figure 8. Composition de la commande NADEM2

On indique le motif du rejet, comme le montre la figure 8.
Signification de la commande : sous-canal non établi.

2.2 Commandes liées aux échanges des données en ligne

2.21 ACC

Nous appelons élément les données destinées à 1 sous-canal transmises lors d'une émission/réception sur la ligne. Cette notion sera précisée par la suite (voir 2.4).

Nous posons comme principe de n'envoyer en ligne de données pour un sous-canal que si le PUITS DE DONNEES s'est déclaré prêt à les recevoir ce qui veut dire en régime établi que l'élément précédent pour ce même sous-canal est parvenu à destination (soit le fichier, soit le tampon s'il s'agit d'un article de fichier fractionné pour les besoins de la transmission, voir 4.324).

La commande ACC joue ce rôle d'accusé de réception positif.

Elle informe que le côté PUITS DE DONNEES est à même de recevoir un nouvel élément, ce qui veut dire que soit il dispose de place mémoire qui vient d'être libérée à la suite d'une écriture correctement effectuée dans un fichier PUITS DE DONNEES, soit que l'élément transmis demandant à être complété, le côté PUITS DE DONNEES attend l'élément suivant sur une longueur inférieure ou égale à celle indiquée.

Dans un cas de réalisation cette longueur peut être implicite.

identifieur de la commande | n° sous-canal | longueur tampon disponible

Figure 9. Composition de la commande ACC.

Signification de la commande : élément précédent bien reçu.

2.22 NACC

Quand la réception de l'élément précédent a débouché sur une anomalie on avertit la SOURCE DE DONNEES par un accusé de réception négatif NACC précisant les raisons de l'échec. (voir figure 10).

identifieur de la commande | n° sous-canal | motif

Figure 10. Composition de la commande NACC

En fonction du motif la SOURCE DE DONNEES pourra soit retransmettre l'élément concerné, soit décréter la suspension de la liaison, la décision étant éventuellement laissée à l'opérateur.

Signification de la commande : condition anormale lors de la réception de l'élément précédent.

2.3 Commandes de suspension des échanges

Elles sont de deux sortes suivant qu'elles concernent un sous-canal donné (RPL) ou l'ensemble des sous-canaux (FERM).

2.31 RPL

identifieur de la commande | n° sous-canal | motif

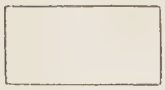
Figure 11. Composition de la commande RPL.

On précise le motif de la rupture du sous-canal pour pouvoir avertir l'opérateur par un message à la console.

L'ordre RPL peut être envoyé soit par la SOURCE DE DONNEES après la réception d'une commande NACC avec un motif mettant fin à la liaison : panne irréparable ou fin de fichier, soit par la SOURCE ou le PUITS DE DONNEES sur ordre extérieur (intervention de l'opérateur).

Signification de la commande : rompre le sous-canal.

2.32 FERM



identifieur
de commande

Figure 12. Composition de la commande FERM

La commande FERM (voyez figure 12, ci-dessus) est envoyée uniquement sur ordre extérieur ; aucun sous-canal ne pourra plus être établi, mais il n'y a pas rupture des sous-canaux déjà existants, il suffit d'attendre les fins de fichier.

Signification de la commande : on va fermer.

2.4 Echanges en ligne - composition du message

Nous comprenons par message l'ensemble des informations que nous voulons émettre en une seule fois ; après adjonction de caractères conformes à la procédure de transmission nous obtenons un bloc de transmission. La composition d'un message est montrée sur la figure 13, ci-dessous :

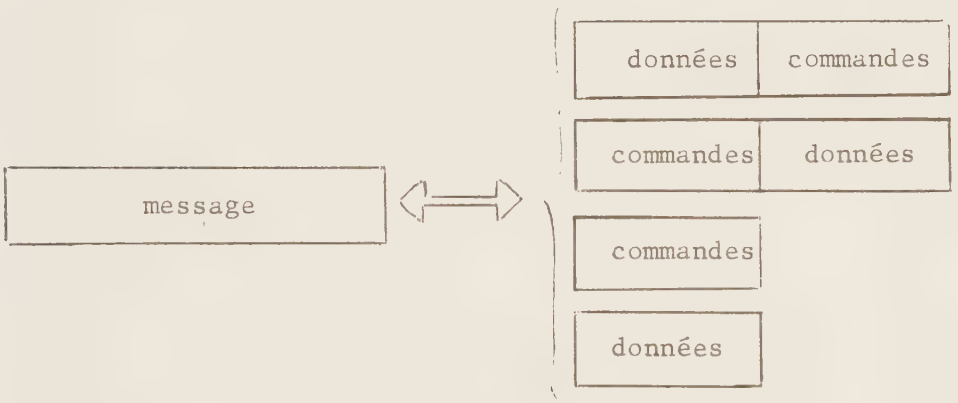


Figure 13. Compositions possibles d'un message

Le message peut comporter deux zones dont l'ordre est sans importance : commandes et données, chacune de ces zones pouvant d'ailleurs être présente seule pour elle-même.

2.41 Zone de commandes

La zone de commandes est représentée sur la figure 14 ci-dessous :

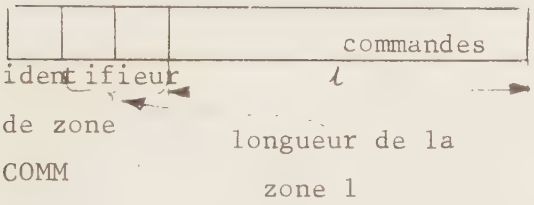


Figure 14 : zone commandes

Elle commence par un identifieur (COMM) suivi de la longueur des commandes placées derrière.

2.42 Zone de données

La zone de données commence (voyez figure 15) elle aussi par un identifieur de zone* (DONN) suivi d'un ou plusieurs éléments.

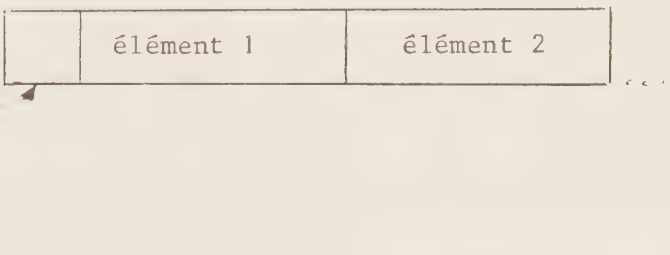


Figure 15. Zone de données

Chaque élément est constitué par l'ensemble des données destinées à un sous-canal et transmises en une seule fois.

---

* On pourrait également considérer que DONN fait partie de la zone de commandes et que sa présence indique qu'une zone de données suit.

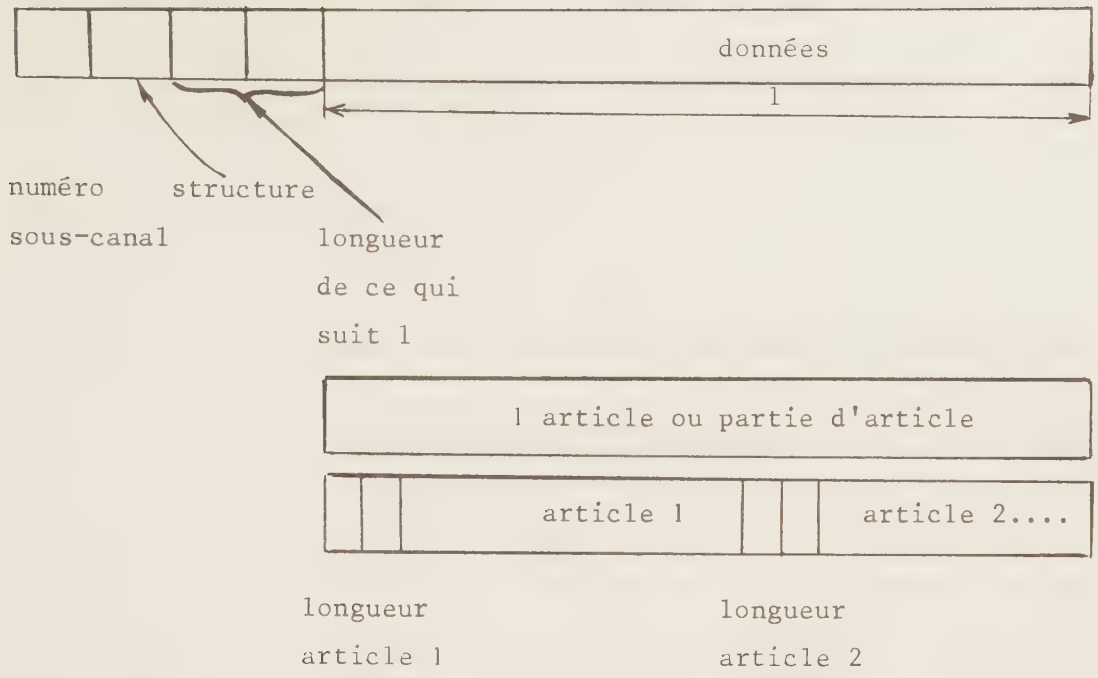


Figure 16. Composition d'un élément

Les relations entre l'élément et le fichier peuvent se résumer en deux points :

1) le fichier est divisé en une suite d'éléments pour la transmission
2) l'ordre des éléments pour la transmission préserve l'ordre dans le fichier.

Un élément peut transporter un article complet, une partie d'article ou plusieurs articles complets (voyez figure 16, ci-dessus). La case "structure" permet de savoir dans lequel de ces cas nous nous trouvons.

Lorsque l'élément comprend plusieurs articles, chacun d'entre eux est précédé par sa longueur.

Cette composition de la zone de données permet de réaliser un multiplexage dans l'espace puisque des données concernant plusieurs sous-canaux peuvent être transmises simultanément.

Un multiplexage temporel (chaque sous-canal à tour de rôle) reste évidemment possible.

Remarquons qu'un caractère de fin de message ou bien une indication de la longueur de la zone de données serait nécessaire uniquement, si l'on ignorait la longueur du message que l'on a reçu sur la ligne.

Il y a un sous-canal qui ne suit pas les règles énoncées ; c'est le sous-canal de communication entre les machines à écrire de pupitre.

Nous considérons qu'il est établi en permanence, il n'y a donc pour lui ni demande d'établissement ni réponse.

D'autre part, nous traitons chaque ligne tapée à la machine à écrire et destinée à être transmise comme un seul article dont la taille ne saurait, de toute évidence, dépasser la longueur d'une ligne de télescriptrice ; dans tous les cas moins de 255 caractères ; un seul octet suffit donc pour indiquer cette longueur.

Tout ceci conduit à adopter pour le sous-canal MAE la structure représentée sur la figure ci-dessous :

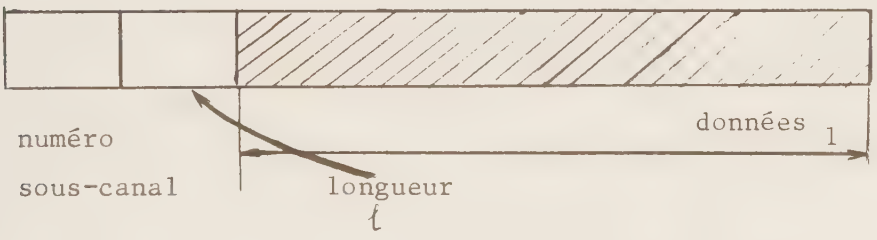


Figure 17. Elément correspondant au sous-canal de communication entre les machines à écrire de pupitre.

3. Etude générale du programme de téléconversion

3.1 Fonctions assurées par le programme

La figure 18, ci-dessous, situe notre programme par rapport au monde extérieur.

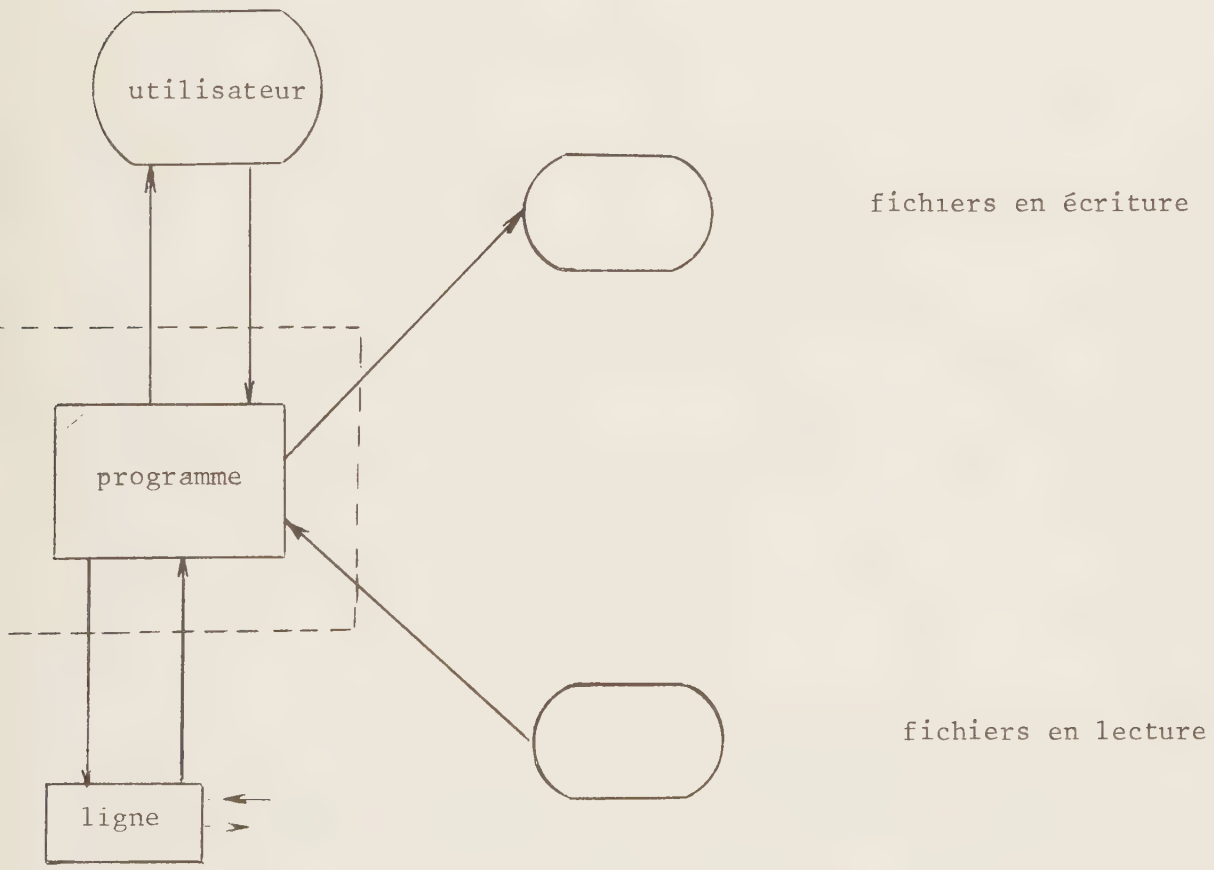


Figure 18

On peut distinguer (voir figure 19) trois grandes "fonctions" parallèles :

- réception (R)
- émission (E)
- communication avec l'utilisateur (racine)

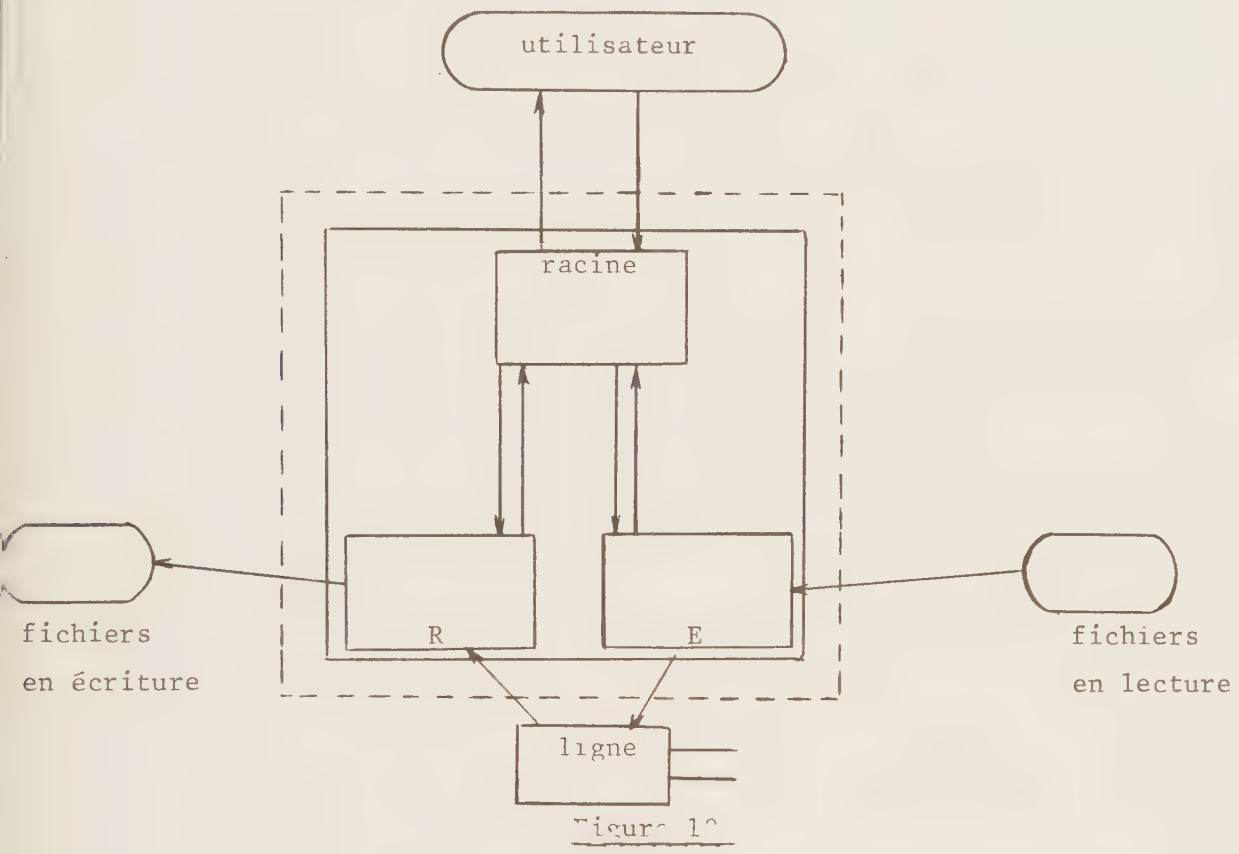


Figure 19

Les fonctions R et E sont en principe indépendantes, mais peuvent se trouver "couplées" dans certaines configurations comme par exemple l'emploi d'une ligne fonctionnant en mode half-duplex ou d'un seul tampon de mémoire servant à l'émission et à la réception.

Une partie du programme doit être en permanence en mémoire des deux côtés, il faut qu'elle soit à même de recevoir et de comprendre une demande de transfert (d'établissement de sous-canal) introduite à la machine à écrire locale ou transmise en ligne par l'ordinateur-correspondant.

Deux solutions sont possibles :

1. incorporer à la racine une fonction séparée s'occupant de la réception des commandes initiales (ou de toutes les commandes) et des éléments du sous-canal MAE et une autre pour l'émission des commandes et des éléments du sous-canal MAE.

2. utiliser des fonctions uniques pour la réception et l'émission des commandes et des données.
   Cette deuxième solution nous semble plus systématique et nous l'adopterons bien qu'elle conduise dans certains systèmes à une augmentation de l'encombrement de la mémoire.

A ce propos, remarquons qu'aucun problème particulier quant à la taille de la mémoire occupée ne se pose dans un système à mémoire paginée gérée en "page à la demande". C'est le cas du système ESOPE sur ordinateur CII 10070 (en cours de réalisation à l'Institut de Recherche d'Informatique et d'Automatique à Rocquencourt) dans lequel la mémoire est allouée seulement au moment de la référence effective à la page.

Dans un système comme SIRIS 2 sur l'ordinateur CII IRIS 50 l'allocation est statique ce qui peut poser certains problèmes et conduire à l'élaboration de solutions particulières.

Nous résumons ci-dessous les fonctions respectives des trois parties que nous avons distinguées dans notre programme :

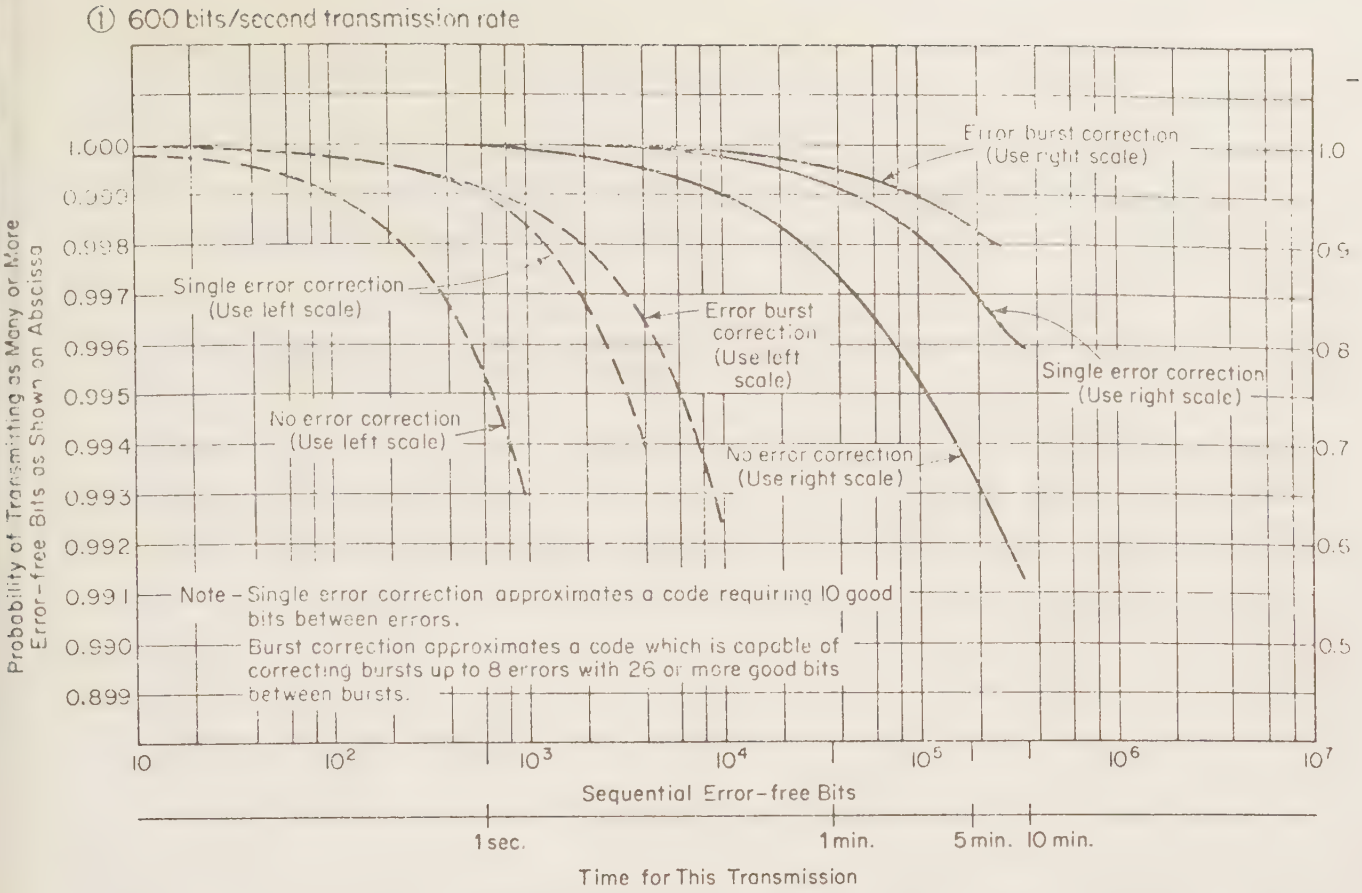
Fonctions de la racine :
- communication avec l'utilisateur par la console opérateur

Fonctions de R :
- réception des données et des ordres sur la ligne
- interprétation des commandes
- écriture dans les fichiers

Fonctions de E :
- lecture dans les fichiers
- émission sur la ligne.

Les messages émis en ligne auront une longueur limitée ; il est souhaitable de la laisser en paramètre dans le programme de façon à pouvoir l'adapter aux caractéristiques de la ligne.

D'une part, on a intérêt à avoir des messages aussi longs que possible car chaque émission/réception sur la ligne prend un certain temps, d'autre part, on ne saurait cependant dépasser une certaine limite à cause des erreurs de transmission dont le nombre croît rapidement avec la longueur du bloc (voir figure 20).



Probabilities of obtaining error-free transmission on voice lines for messages of different lengths, taken from measurements made on AT&T's switched public network. *Redrawn from CCITT Red Book, Vol. VII, published by International Telecommunication Union, Geneva, March 1961.*

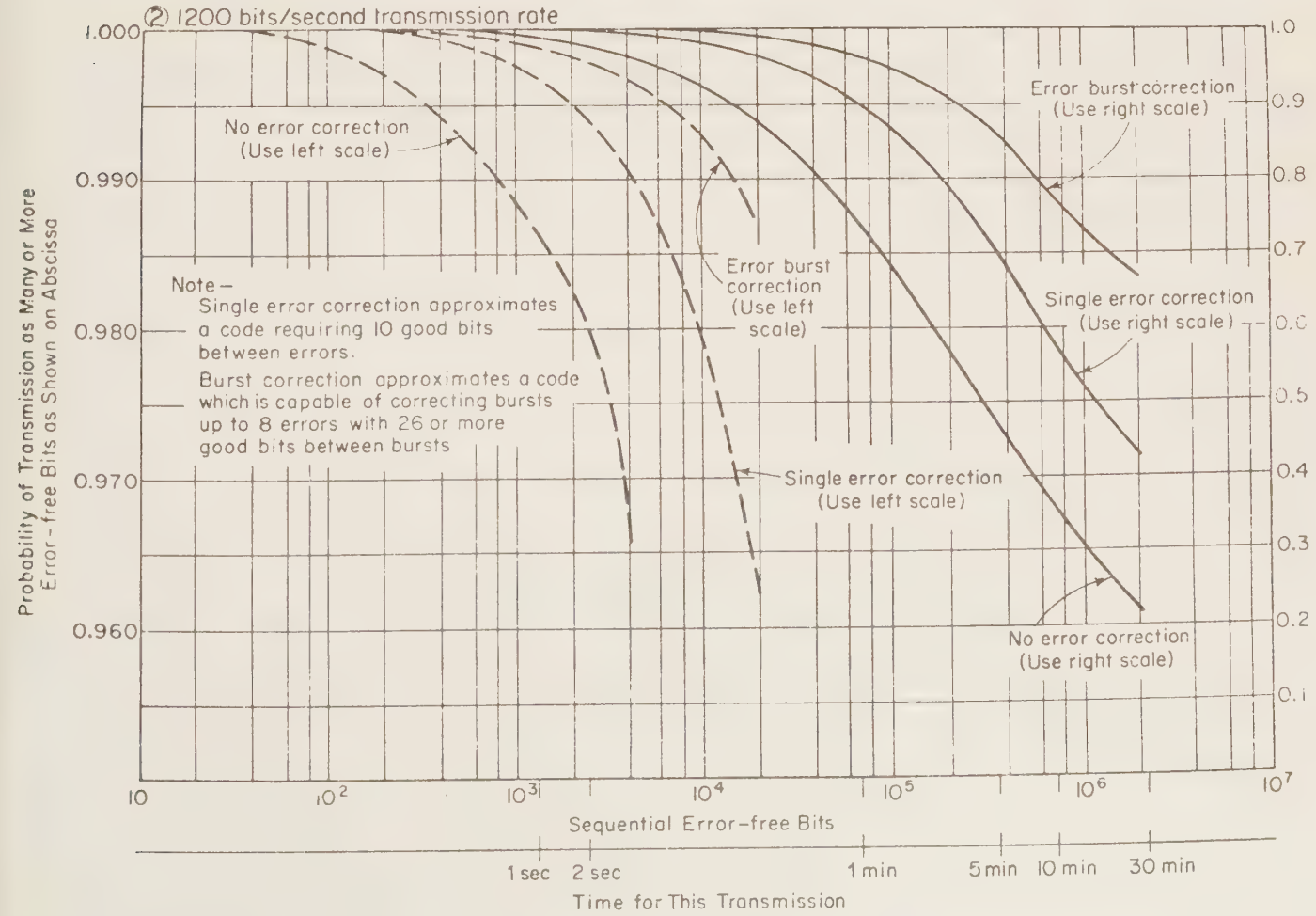


Figure 20

Il y a, par conséquent, un optimum à trouver qui dépend du taux d'erreurs sur la ligne et de l'intervalle de temps indispensable entre blocs.

Nous ne nous occuperons pas de la détection et de la correction des erreurs ; remarquons simplement que malgré l'existence de codes correcteurs d'erreurs perfectionnés, il peut être préférable de recourir à l'emploi d'un code détecteur d'erreurs avec retransmission de messages erronés.

Ceci parce que plus grande est l'efficacité d'un code correcteur plus bas est son rendement en caractères utiles (50 % par exemple pour le code Hagelbarger).

3.2 Organisations possibles du programme

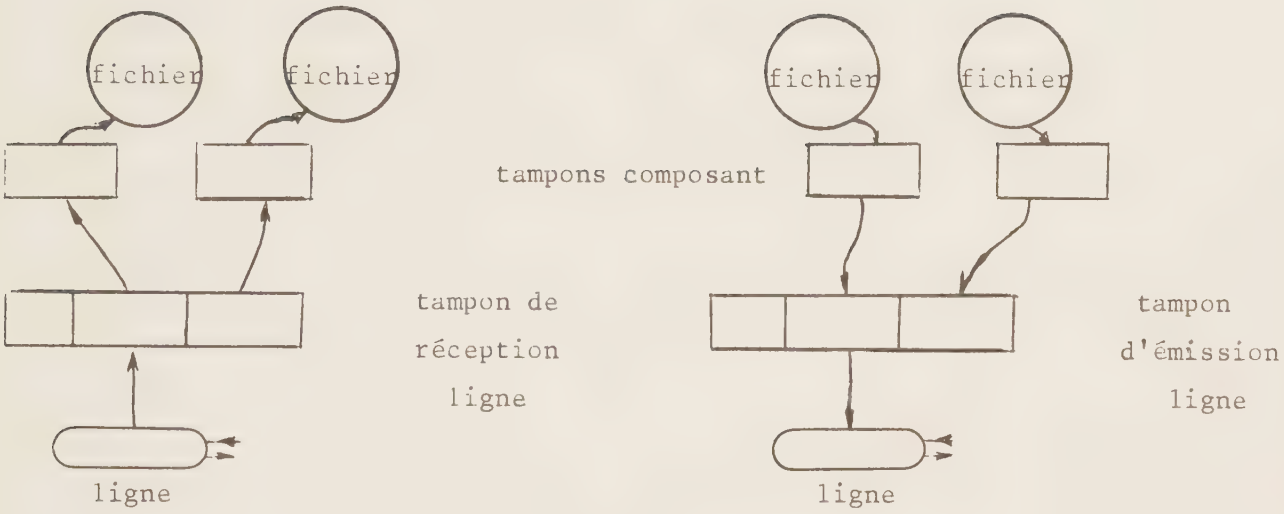
En toute généralité, la fonction "réception" doit disposer de (figure 21 ci-dessous) :

- un (ou plusieurs) buffer de réception ligne dans lequel on reçoit les messages en provenance de l'ordinateur-correspondant,
- tampons composant pour les différents sous-canaux, les articles de fichier seraient reconstitués dans ces tampons de mémoire. Chacun d'entre eux serait adapté à la taille maximum de l'article.

De façon analogue pour la fonction "émission" nous devons avoir :

- un tampon composant par sous-canal SOURCE adapté à la taille maximum de l'article du fichier lu,
- un (ou plusieurs) buffer d'émission ligne servant à l'émission des messages.

Ces tampons, bien entendu, n'existent pas forcément tous en réalité mais il nous a semblé bon de présenter les choses de cette façon. Nous allons maintenant passer en revue les différentes organisations possibles en nous référant au schéma de la figure 21.



a. fonction "réception"

b. fonction "émission"

Figure 21. Organisation générale

3.21 Fonction "émission"

1ère solution :

- allouer effectivement à chaque sous-canal SOURCE un tampon composant
- réserver un tampon émission ligne.

Avantages : solution simple, gestion simple : on vide les tampons composant pour former des éléments dans le tampon émission ligne.
Permet très facilement de lire par anticipation l'article suivant dans le fichier en le plaçant dans le tampon composant.

Inconvénients : demande beaucoup de place mémoire, impose des transferts de chaînes de caractères (recopies de tampons) qui sont des opérations longues.

2ème solution :

- émettre directement à partir des tampons composant (c'est-à-dire supprimer le tampon émission ligne en gardant toujours un tampon composant en entrée par sous-canal SOURCE).

Avantages : exige moins de place mémoire que la solution 1, est également plus rapide que celle-ci puisqu'on évite, en partie au moins, les recopies de tampons.

Inconvénients : une difficulté apparaît pour la zone de commandes et le sous-canal de communication entre les machines à écrire de pupitre ; il faut créer un tampon auxiliaire dans lequel on formera les commandes à envoyer.

Cette solution a le plus d'intérêt si l'on peut chaîner les informations dans les commandes de canal pour l'envoi en ligne.

3ème solution :

- on convient de multiplexer les sous-canaux dans le temps uniquement, si le chaînage de données pour l'émission en ligne n'est pas possible :
- créer plusieurs tampons émission ligne dans lesquels on lira les articles (qui serviront donc également de tampons composant), voir figure 22 ci-dessous.

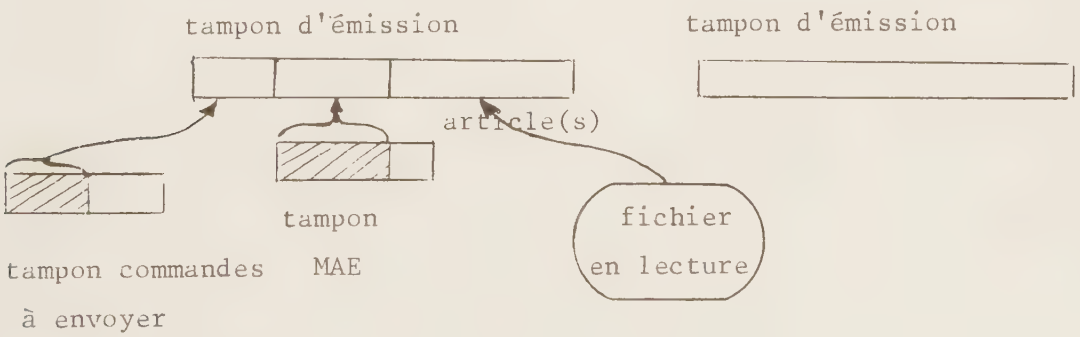


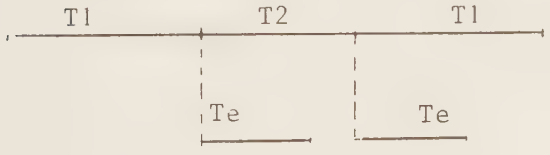
Figure 22.

On pourrait se limiter à deux tampons d'émission travaillant en bascule. Malheureusement, ce mécanisme se trouverait faussé dans le cas d'articles de taille supérieure à la taille du message car aucune nouvelle lecture ne serait alors possible avant le vidage complet du tampon.

Il est facile de voir que l'on atteint l'optimum (le minimum de ralentissement) avec ce système, si la taille des articles reste inférieure à la longueur du message, lorsque l'invervalle de temps qui sépare deux émission consécutives sur la ligne est inférieur ou égal au temps de remplissage du tampon (figure 23).

T1 temps de garnissage du tampon 1
T2 temps de garnissage du tampon 2
Te temps qui sépare deux émissions consécutives sur la ligne

a) temps d'émission supérieur au temps de garnissage



b) temps d'émission inférieur au temps de remplissage

Figure 23. Emploi de deux tampons d'émission.

3.22 Fonction "réception"

Là aussi s'offrent à nous plusieurs solutions correspondant souvent à des degrés de souplesse différents.

Ainsi, si nous autorisons le multiplexage dans l'espace (présence simultanée dans le message de plusieurs éléments) et si, de surcroît, le nombre et l'ordre des éléments nous est inconnu nous pouvons adopter la solution qui consiste à avoir un tampon de réception ligne et autant de tampons composant qu'il y a de

sous-canaux PUITS. Le principal avantage de cette solution est que les périphériques lents ne freinent pas les périphériques rapides. Elle est particulièrement bien adaptée à un raisonnement en termes de processus parallèles.

On procéderait alors comme suit (figure 24) :

après une analyse du message reçu on " réveillerait" les processus :

- décodage (interprétation)des commandes
- sortie.....
- sortie.....

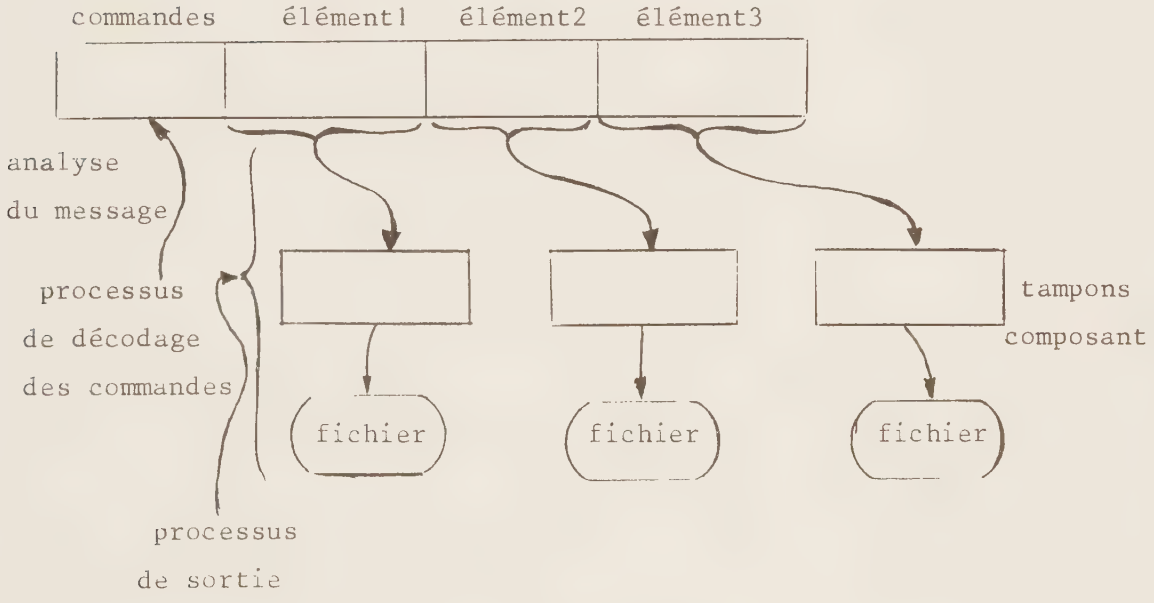


Figure 24.

Chaque processus de sortie assure le transfert de l'élément dans le tampon composant et, si l'article n'est pas incomplet, son écriture dans le fichier. L'inconvénient de cette solution est la place mémoire qu'elle demande. Si pour y parer nous optons pour un nombre de tampons inférieur au nombre de sous-canaux PUITS,la gestion s'en

trouvera notablement compliquée, du seul fait que la réception d'un ACC pour un sous-canal donné ne nous autorisera plus, par elle-même, à envoyer l'élément suivant.

Au cas où nous acceptons exclusivement le multiplexage dans le temps, nous nous attendons à voir au maximum une zone de commandes, un élément MAE et un élément d'un autre sous-canal. Nous pouvons essayer de supprimer le tampon de réception ligne et de recevoir directement dans les tampons composants.

La principale difficulté dans la mise en oeuvre de cette solution vient des articles qui ont dû être segmentés pour la transmission (ce problème ne se poserait pas si l'on ne considérait que des fichiers spécialement créés pour la transmission, dans lesquels on limiterait la longueur des articles, de façon à la rendre inférieure à la longueur du message transmissible en ligne).

On sera amené à effectuer des recopies de tampons à cause de la zone de commandes et MAE. Pour chaque tampon il faudrait réserver une longueur l telle que l = lmax + laux, où

lmax est la longueur maximum d'article

et

laux désigne la longueur de la zone commandes et de l'élément du sous-canal MAE.

On ne gagne en mémoire occupée que jusqu'à un certain nombre k de sous-canaux PUITS. Ce nombre pourrait être déterminé approximativement de la relation $k * laux \leqslant$ 1 tampon ligne

$$k \leqslant \frac{\text{longueur tampon ligne}}{laux}$$

Une autre solution consiste à avoir 1 tampon de réception ligne et plusieurs (mais moins que de sous-canaux) tampons composants. Elle est motivée par le fait qu'un seul tampon composant est utilisé lors de chaque réception (dans le cas du multiplexage dans le temps seulement).

Les tampons composant ne seraient pas liés de façon fixe à des composants donnés, mais serviraient en bascule aux différents sous-canaux.

Une difficulté apparaît pour les articles fragmentés car le buffer se trouve alors bloqué tant que l'article n'a pas été reconstitué.

Il faut, par conséquent,transmettre les parties d'article les unes après les autres en priorité sur d'autres sous-canaux et s'imposer en plus la règle évidente que le nombre de sous-canaux PUITS à articles fragmentés ne dépasse pas le nombre de tampons composant .

Mentionnons pour terminer une solution extrémale avec fonctionnement à un seul tampon utilisé à la fois pour la réception et la sortie dans le fichier. Le recours à cet arrangement serait justifié dans le cas où le temps de transmission serait long comparé au temps d'écriture dans le fichier.

Cette énumération des solutions possibles qui, d'ailleurs, ne prétend pas être exhaustive, confirme encore une fois la largeur de la marge de manoeuvre si souvent invoquée dans le domaine de l'informatique dès qu'il s'agit d'annoncer des préférences nettement arrêtées et met en évidence la part, encore toujours importante, d'arbitraïre dans le choix des structures.

3.3 Structure hiérarchisée

3.31 Interfaces standard

Nous considérons ici une approche légèrement différente du problème caractérisée par le souci de définir le maximum d'interfaces standardisés.

Une conversion de support consiste, comme son nom l'indique, en un changement de milieu servant de support au fichier.

A cette fin, nous devons disposer de composants pouvant manier les milieux, à chacun des composants étant associé un dispositif qui permet d'effectuer la fonction de transfert entre le système de fichiers et le composant.

Appelons device cet ensemble. La figure 25 montre alors le schéma d'une conversion de support ; D et D' sont les devices.

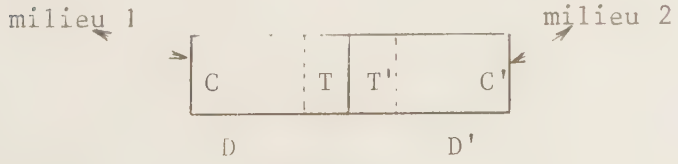


Figure 25.

L'interface entre D et D' est défini par le système de fichiers (la forme d'un article).
Dans une conversion locale un certain nombre d'informations transite à travers cet interface ; ces informations peuvent être divisées de façon générale en données du fichier et un répertoire de commandes.

S'il s'agit d'une téléconversion de support les deux devices sont physiquement séparés. Cette séparation introduit un nouvel aspect en ce qui concerne le chemin entre D et D'. Nous ne voulons pas que l'interface qui existait précédemment entre eux s'en trouve affecté. Ceci va nous conduire à introduire entre D et D' une fonction de transfert dont le seul but est de permettre aux deux devices de faire abstraction du fait qu'ils sont physiquement distants et de se comporter comme s'ils étaient l'un à côté de l'autre ; la situation est représentée sur la figure 26 ci-dessous.

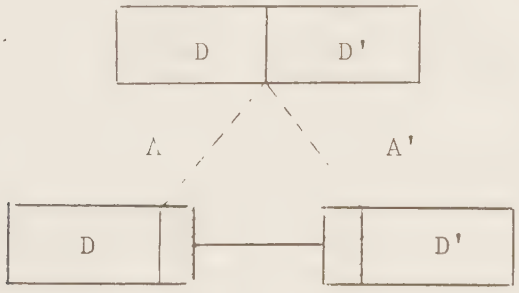


Figure 26.

Remarquons que les interfaces aux points A et A' sont identiques vus à partir de D et de D' puisque la seule fonction des parties hachurées est de transmettre un flux d'information entre D et D'.

Il est bon de rendre cet interface standard.

Ceci permet d'assembler plusieurs D's et D''s autour d'une fonction de transfert commune comme le montre la Figure 27.

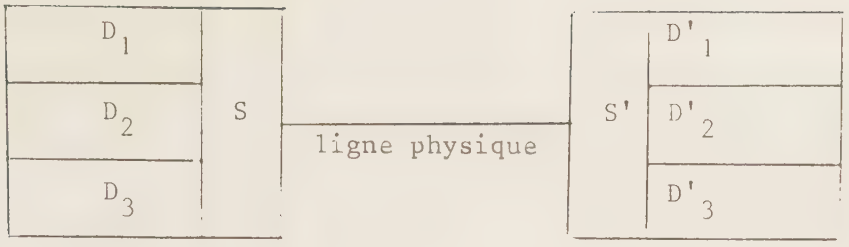


Figure 27.

De cette façon plusieurs D's et D''s peuvent être couplés à travers une seule ligne physique. La fonction de transfert prend le nom de station (S et S'). Il faut souligner que S et S' ne sont pas tout à fait les mêmes que la fonction de transfert de la figure 26 car on doit maintenant en plus assurer la gestion de l'accès des différentes paires D-D' à la ligne physique.

Jusqu'à présent nous n'avons pas réellement fait intervenir la ligne physique dans la discussion. Il serait intéressant de rendre S et S' indépendants du type de ligne employé et d'en séparer les fonctions du niveau ligne proprement dit. On obtient alors le schéma de la figure 28.

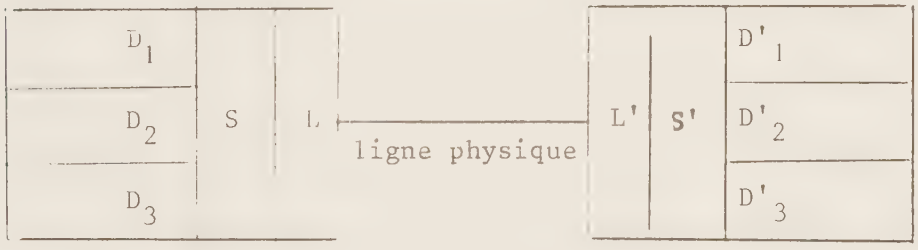


Figure 28.

Comme le nouvel interface défini est le même pour toutes les stations nous pouvons en toute généralité connecter plusieurs stations à la partie qui dépend de la nature de la ligne (L et L') ce qui conduit au schéma de la figure 29.

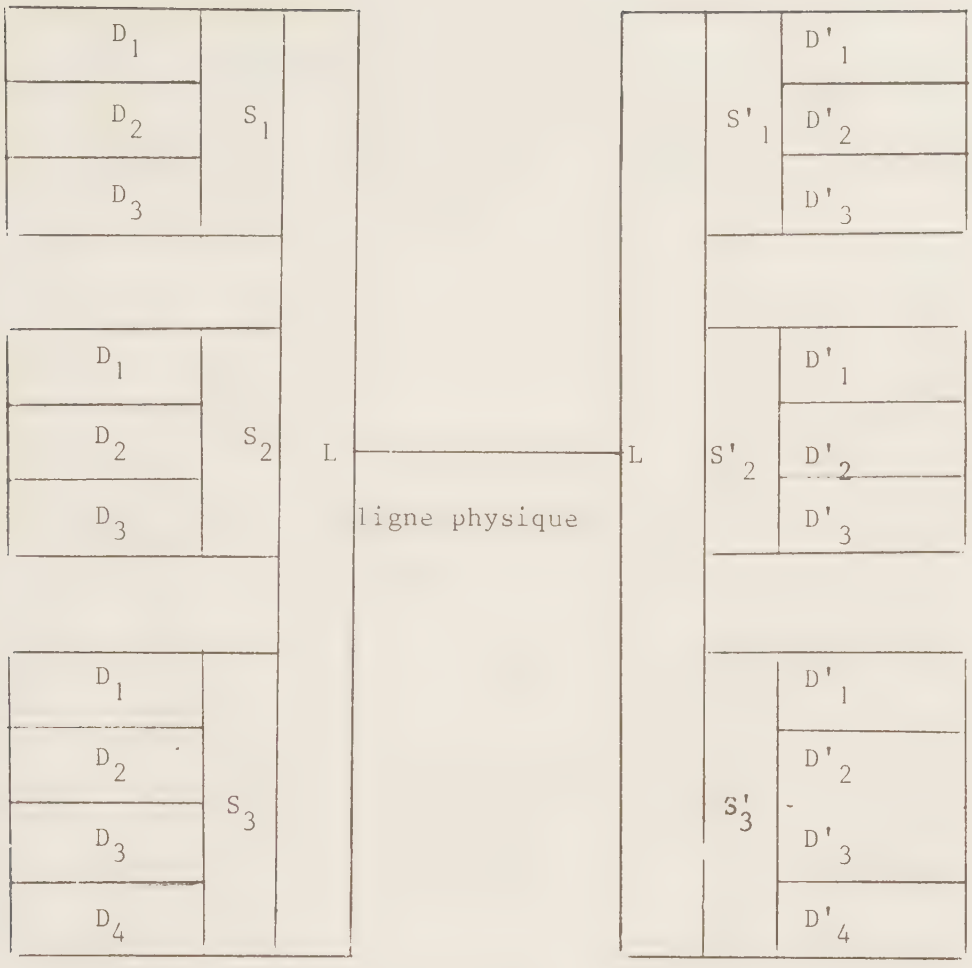


Figure 29.

3.32 Liaisons conceptuelles et niveaux

De cette façon, on peut distinguer un niveau device comprenant D et D' connectés par une liaison conceptuelle qui leur permet de communiquer tout à fait comme s'ils étaient physiquement adjacents.

Ce lien conceptuel est créé grâce au niveau station dont la seule raison d'être est de faire de façon à ce que la liaison conceptuelle paraisse réelle à chaque niveau device dont la station est responsable .

Ainsi D et D' sont liés conceptuellement au moyen de leurs interfaces avec le niveau station, et S et S' de la même manière sont liés conceptuellement du fait de la présence du niveau ligne, et L et L' prennent en charge la transmission d'information sur la ligne physique.

Nous sommes arrivés à définir dans ce qui précède une structure hiérarchisée symétrique par rapport à la ligne physique comme on peut le voir sur la figure 30.

Pour la définition des commandes échangées sur les différentes lignes conceptuelles on peut adopter la tactique qui consiste à considérer que l'on fait abstraction des niveaux inférieurs et l'ensemble des commandes ainsi définies sera le même que si les deux parties étaient physiquement adjacentes.

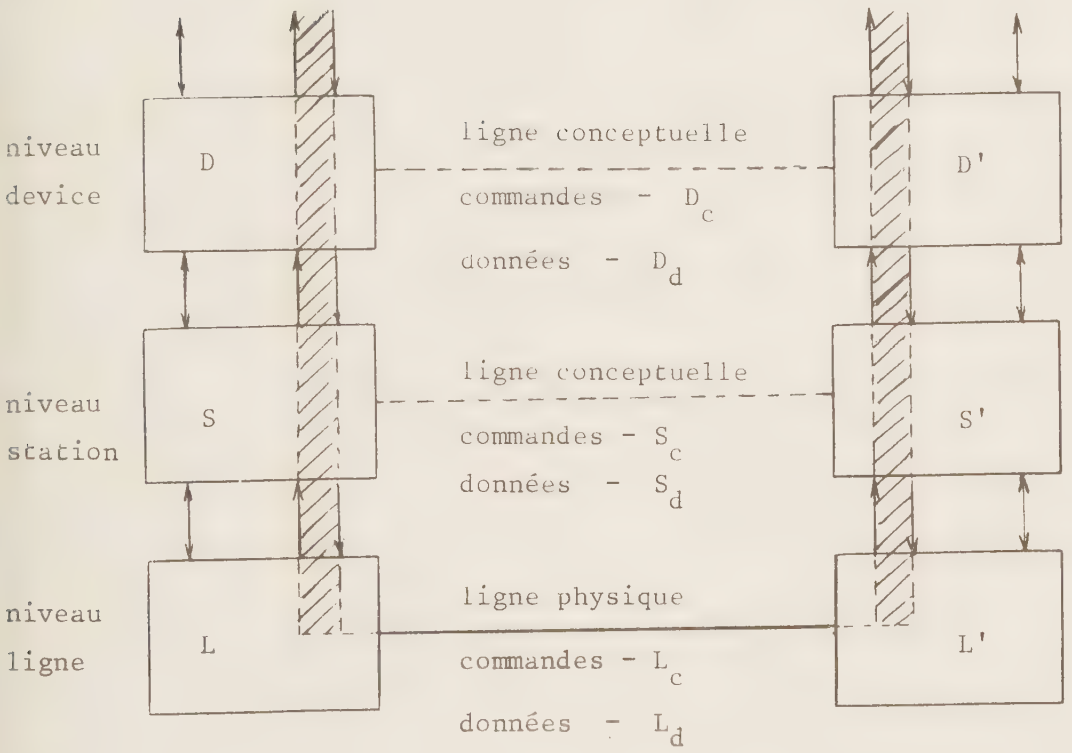


Figure 30.

Les interfaces entre les différents niveaux peuvent être représentés en termes d'ordres qui symbolisent les effets voulus. Certains de ces ordres sont explicites, d'autres pas.

D'une façon générale et pas très précise on peut définir ces interfaces comme ce qui est nécessaire pour la transmission des données entre les niveaux qu'ils séparent.

Les commandes effectivement échangées en ligne s'adressent à plusieurs niveaux à la fois.

Au début de notre étude à la Compagnie Internationale pour l'Informatique nous avions adopté cette façon de voir la liaison UC-UC comme une structure à niveaux où chaque niveau correspond à une étape dans le cheminement de l'information.

Nous avions alors distingué un niveau supplémentaire entre les niveaux device et station. Ce niveau portant le nom de "transport élément" permettait le passage d'articles de fichier aux éléments pour les différents sous-canaux.

Les fonctions des différents niveaux étaient délimitées de la façon suivante :

- le niveau ligne permet d'échanger des blocs contenant des ordres et des données
- le niveau station prépare les messages à présenter au niveau ligne.
  Il assure, à l'émission, le transfert des informations du (ou des) tampon composant dans le tampon d'émission d'après certaines règles de priorité, ou de façon générale gère le choix du ou des sous-canaux dont l'élément doit faire partie du message.
  A la réception, il prépare le transfert du tampon réception ligne au tampon composant.
- le niveau transport élément, à l'émission, gère le remplissage des tampons composant .
  A la réception, il participe au garnissage du tampon composant.

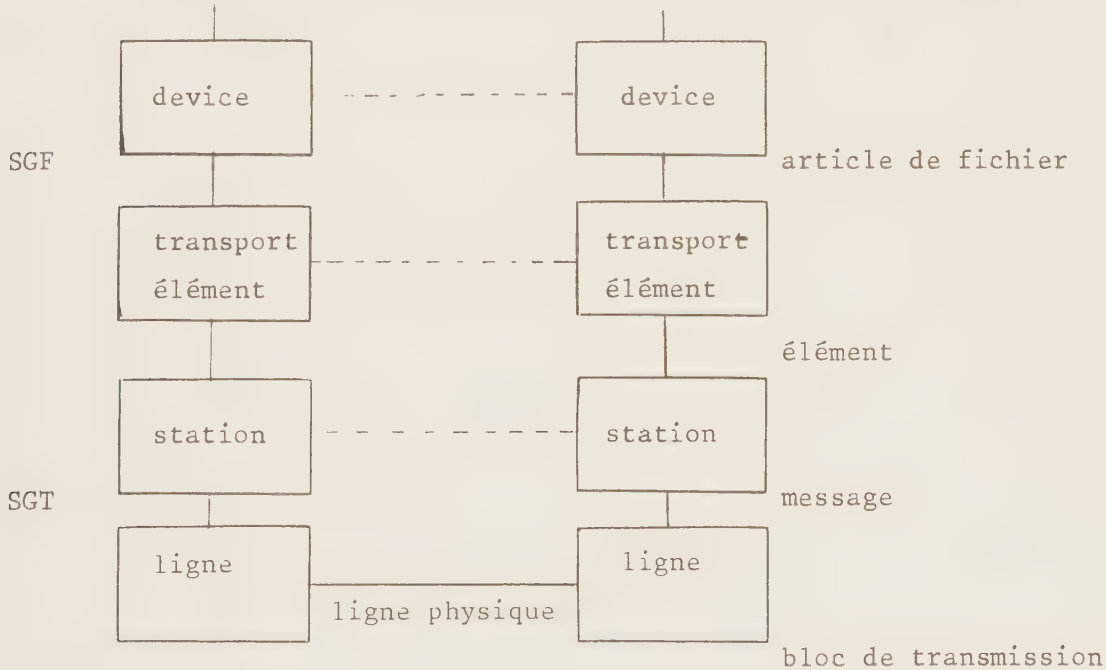


Figure 31.

Sur la figure 31 nous avons indiqué la forme que prenait l'information à transmettre à chaque interface ainsi que la position du système de gestion de fichiers (SGF) et du système de gestion de transmissions (SGT).

Cette approche ne préserve malheureusement pas du grand arbitraire dans le partage des fonctions entre niveaux et ne permet pas d'éviter que des considérations pratiques prennent le dessus sur les principes ; son avantage est que grâce aux interfaces standardisés l'extension du système est rendue facile.

4. Application à un système hypothétique.

4.1 Définition du système

Nous tâcherons maintenant de développer une solution dans un système hypothétique que nous supposons muni des traits suivants :

1) il est possible de chaîner les informations pour l'émission en ligne,
2) l'allocation de la mémoire s'effectue d'après le principe de la page à la demande,
3) un usager a la possibilité de créer des processus parallèles
4) un usager peut créer et utiliser des sémaphores.

Ces deux dernières notions sont comprises dans le sens qui leur a été donné par Dijkstra dans " Co-operating Sequential Processes". [6] Rappelons brièvement que les seules opérations autorisées sur un sémaphore au nombre de quatre sont :

La création d'un sémaphore, sa destruction et les opérations V et P dont nous rappelons les définitions respectives en citant Dijkstra :

opération V : une opération à 1 argument lequel est obligatoirement un sémaphore ; sa fonction est d'augmenter de 1 la valeur de son argument. Cette augmentation doit être considérée comme une opération indivisible.

opération P : une opération à 1 argument lequel doit obligatoirement être un sémaphore ; sa fonction est de diminuer de 1 la valeur de son argument dès que le résultat serait non négatif. La décision que le moment est venu pour effectuer cette diminution et la diminution elle-même sont à considérer comme une opération indivisible.

L'opération P représente une attente potentielle : en effet, si un processus effectue une opération P sur un sémaphore qui à ce moment vaut 0,l'opération P ne peut pas être terminée tant qu'un autre processus n'a pas effectué une opération V sur le même sémaphore en lui donnant ainsi la valeur 1.

4.2 Remarques sur la synchronisation des processus à l'aide de sémaphores.

Le sémaphore muni de ces 2 opérations (P et V) est un excellent outil d'exclusion mutuelle. Il nous semble cependant très "fort" et très "sévère" comme moyen de synchronisation de processus.

Une opération P prématurée conduit inéluctablement à la suspension du processus.

C'est sans doute la raison qui a conduit dans certains systèmes à l'introduction d'opération du type : "tester la valeur d'un sémaphore". Ce genre d'opération nous paraît contraire à l'esprit de la définition du sémaphore qui stipule bien que les seules façons d'accéder à un sémaphore sont les opérations P et V.

Souvent dans les programmes il est nécessaire de synchroniser un processus avec l'arrivée d'un événement.

4.21 Evénements simples

S'il s'agit d'un événement simple et si nous sommes sûrs qu'il arrivera, il suffit que le processus qui l'attend effectue une opération P. Nous créons un sémaphore événement arrivé avec la valeur initiale 0.

Le processus 1 est le processus qui attend l'événement, le processus 2 assure l'arrivée de l'événement.

```
"begin integer
      parbegin
      processus 1 : begin
                      :
                    P(événement arrivé) ;
                    traitement lié à l'arrivée de l'événement ;
                      :
                    end ;
      processus 2 : begin
                      :
                    V(événement arrivé) ;
                      :
                    end ;
      parend
end"
```

Mais si jamais on n'est pas certain que l'événement se produira,il faut s'interdire l'emploi de ce genre de séquence.

Ce qu'on peut faire dans ce cas c'est de définir un drapeau (flag) en en protégeant l'accès par un sémaphore. On a montré ci-dessous un exemple d'un traitement aiguillé en fonction de l'arrivée d'un événement :

```
"begin integer flag, accès au flag ;
 accès au flag := 1 ;
 flag := 0 ;
      parbegin
      processus 1 : begin
                      :
                    P(accès au flag) ;
                    if flag = 1 then
                       begin
                       flag := 0 ;
                       traitement 1 ;
                       end
                    else traitement 2 ;
                    V(accès au flag) ;
                       :
                    end ;
      processus 2 : begin
      événement   : P(accès au flag) ;
                    flag 1 := 1 ;
                    V(accès au flag) ;
                    :
                    end
      parend
end"
```

4.22 Evénements composés

4.221 Intersection des événements

L'événement attendu peut être une combinaison logique de plusieurs événements. La synchronisation d'un processus sur l'arrivée d'un événement

composé (événement 1 arrivé) ∩ (événement 2 arrivé) ∩.....∩ ne présente pas de difficultés particulières.

Nous définissons autant de sémaphores qu'il y a d'événements composants avec des valeurs initiales nulles. Après, il suffit d'effectuer une séquence d'opérations P l'une à la suite de l'autre portant sur les sémaphores définis :

```
"begin integer événement 1 arrivé, événement 2 arrivé, événe-
              ment 3 arrivé ;
 événement 1 arrivé := 0 ;
 événement 2 arrivé := 0 ;
 événement 3 arrivé := 0 ;
      parbegin
      processus 1 : begin
                      .
                      .
                    P(événement 1 arrivé) ;
                    P(événement 2 arrivé) ;
                    P(événement 3 arrivé) ;
                    traitement ;
                      .
                      .
                    end ;
      processus 2 : begin
                      .
                      .
                    V(événement 1 arrivé) ;
                      .
                      .
                    end ;
      processus 3 : begin
                      .
                      .
                    V(événement 3 arrivé) ;
                      .
                      .
                    end ;
      parend
 end"
```

L'ordre des opérations P dans le processus 1 est sans importance puisque tout ce qui nous intéresse est la réalisation de l'ensemble des trois conditions.

4.222 Réunion des événements

Les choses ne sont pas aussi simples lorsque nous voulons mettre un processus en attente d'un événement parmi n.

Nous ne pouvons en aucun cas initier une suite d'opérations P puisqu'il nous est impossible de prévoir quel sera l'événement arrivé.

Nous pouvons songer à définir un sémaphore commun avec la valeur initiale 0 et chaque fois qu'un événement parmi les événements attendus est arrivé,à effectuer une opération V sur ce sémaphore :

```
"begin integer sémaphore commun ;
 sémaphore commun := 0 ;
      parbegin
      processus 1 : begin
                      :
                    P(sémaphore commun) ;
                    traitement ;
                      :
                    end ;
      processus 2 : begin
                      :
      événement 1 : V(sémaphore commun) ;
                      :
                    end ;
      processus 3 : begin
                      :
      événement 2 : V(sémaphore commun) ;
                      :
                    end ;
      parend
 end".
```

Remarquons qu'il est utile de garder quelque part l'identité des événements arrivés. Ceci peut se faire de bien des façons par exemple en positionnant des drapeaux correspondants aux événements.

La façon de procéder que nous venons de représenter n'est pas toujours satisfaisante comme nous le montrera le cas de la figure 32.

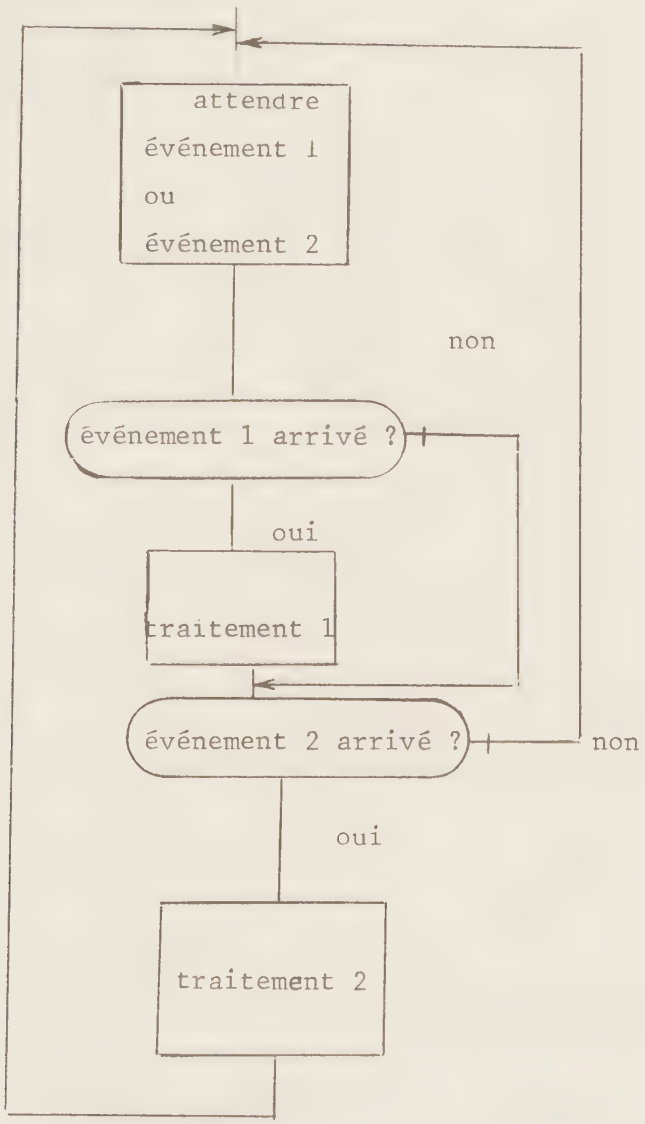


Figure 32.

Faisons correspondre le drapeau flag 1 à l'événement 1 et le drapeau flag 2 à l'événement 2 de la façon suivante :

flag 1 = 0 événement 1 non arrivé
flag 1 = 1 événement 1 arrivé
et même chose pour flag 2 et événement 2.

Le programme devient alors :

```
"begin integer sémaphore commun, flag 1, flag  2 ;
 sémaphore commun := 0 ;
 flag 1 := 0 ;
 flag 2 := 0 ;
       parbegin
       processus 1 : begin
       attente :     P(sémaphore commun) ;
                     if flag 1 = 1 then
                        begin
                        flag 1 := 0 ;
                        traitement 1 ;
                        end ;
                     if flag 2 = 1 then
                        begin
                        flag 2 := 0 ;
                        traitement 2 ;
                        end ;
                     goto attente ;
                     end ;
       processus 2 : begin
                      :
       événement 1 : flag 1 := 1 ;
                     V(sémaphore commun) ;
                      :
                     end ;
       processus 3 : begin
                      :
       événement 2 : flag 2 := 1 ;
                     V(sémaphore commun) ;
                      :
                     end ;
       parend
end".
```

Le programme n'est pas tout à fait correct* ; il peut arriver, en effet, que le processus 1 puisse terminer son opération P sur le sémaphore commun sans qu'il y ait effectivement un traitement quelconque à effectuer ; il suffit pour cela que les deux processus 2 et 3 effectuent les opérations V sur sémaphore commun presque simultanément, par exemple que l'événement 2 arrive pendant le traitement de l'événement 1.

Nous montrons ci-dessous une solution libre de ce défaut ; à chaque condition réalisée nous associons une opération P sur le sémaphore commun ce qui fait qu'au moment où on va en "attente" la valeur de sémaphore commun est égale au nombre d'événements arrivés et non traités.

```
"begin integer sémaphore commun, flag 1, flag 2 ;
 sémaphore commun := 0 ;
 flag 1 := 0 ;
 flag 2 := 0 ;
       parbegin
       processus 1 : begin
       attente :     P(sémaphore commun) ;
                     V(sémaphore commun) ;
                     if flag 1 = 1 then
                        begin
                        flag 1 := 0 ;
                        traitement 1 ;
                        P(sémaphore commun) ;
                        end ;
```

---

* Nous supposons que les événements 1 et 2 ne peuvent pas arriver à nouveau tant que le processus 1 n'a pas terminé respectivement les traitements 1 et 2.

```
                    if flag 2 = 1 then
                       begin
                       flag 2 := 0 ;
                       traitement 2 ;
                       P(sémaphore commun) ;
                       end ;
                    goto attente ;
                    end ;
          processus 2 : begin
                        :
                        :
                    end ;
                        :
                        :
          parbegin
    end".
```

Nous donnons en annexe une justification de cette solution.

4.3 Description du programme

Nous supposons que la gestion des fichiers est assurée par le système et nous laisserons de côté tous les problèmes liés à l'accès à ces derniers.

Trois grandes fonctions ont été distinguées ; nous les préciserons l'une après l'autre.

4.31 Racine

Elle doit permettre la communication avec l'opérateur (l'utilisateur); on peut la réaliser sous forme de deux processus dont l'un s'occupera de la réception et de l'interprétation des messages introduits à la MAE et l'autre de l'émission des messages vers l'opérateur. Considérons d'abord le processus de sortie sur la MAE.

4.311 Processus de sortie sur la machine à écrire

Les adresses des messages à émettre seront consignées dans une table qui contiendra au début l'adresse du message initial faisant savoir à l'utilisateur que le programme est en place. L'accès à la table est partagé avec tous les processus désirant faire émettre un message ; l'exclusion mutuelle sera assurée par le sémaphore accès adresse message.

Cette table n'aura évidemment qu'une taille limitée et, pour éviter les risques de débordement, nous introduirons le sémaphore nombre de messages.

Le processus est activé par l'intermédiaire du sémaphore message à émettre. Tout processus ayant un message à faire émettre sur la MAE devra exécuter la séquence suivante :

```
P(nombre de messages) ;
P(accès adresse message) ;      l'ordre de ces 2 opérations est
                                    essentiel
        :
inscription adresse dans la table
        :
V(accès adresse message) ;
V(message à émettre) ;
```

L'ordre des deux opérations P est essentiel car on risque de bloquer aussi bien le processus qui effectue cette séquence que le processus de sortie. L'algorithme de ce dernier est représenté sur la figure 33.

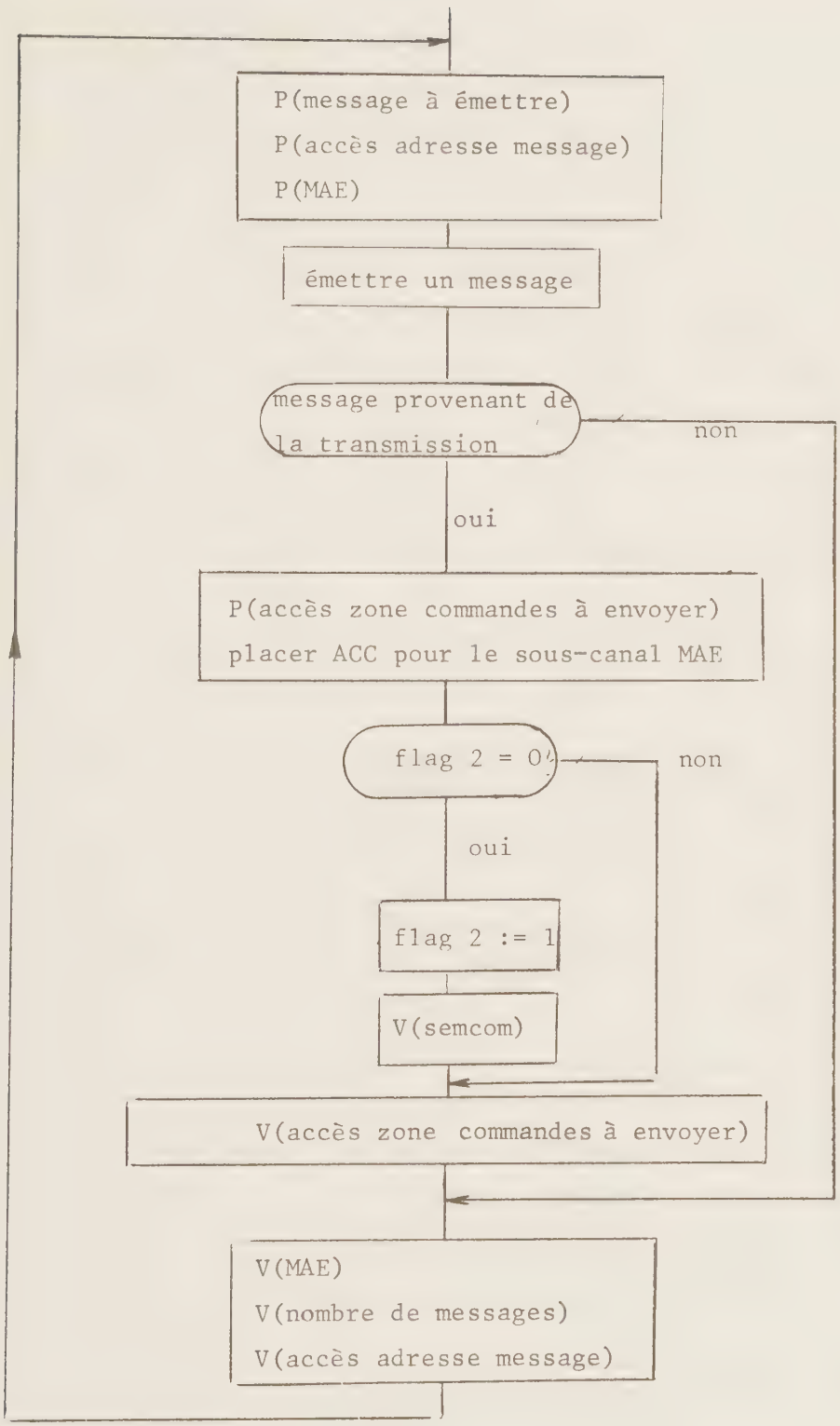


Figure 33 algorithme du processus de sortie sur la machine à écrire de pupitre. Le sémaphore MAE symbolise l'exclusion mutuelle dans l'accès à la machine à écrire.

Si le message que l'on vient de sortir provenait de la transmission sur ligne on prépare un accusé de réception ACC à envoyer ; il sera placé dans une zone spéciale appelée zone commandes à envoyer dont l'accès est protégé au moyen d'un sémaphore. Le drapeau flag 2 est positionné lorsqu'il s'agit de la première commande placée dans la zone ; c'est alors également qu'une opération V est effectuée sur le sémaphore semcom.

En résumé, le processus utilise la table des adresses à émettre, la zone des commandes à envoyer et six sémaphores :

- accès adresse message initialisé à la valeur : 1
- message à émettre initialisé à la valeur : 1
- MAE initialisé à la valeur : 1
- nombre de messages initialisé à la valeur : taille de la table -1,
- accès zone commandes à envoyer initialisé à la valeur : 1
- semcom initialisé à la valeur : 0.

4.312 Processus de réception et d'interprétation des messages MAE.

Les messages tapés à la machine à écrire seront reçus dans un buffer spécial.
Le premier caractère du message permettra de savoir s'il s'agit d'une commande ou d'un élément de transmission.
Dans ce dernier cas le drapeau flag 1 sera positionné, ce qui signalera au processus de formation du message à être transmis en ligne la présence d'un élément MAE dans le buffer. Tant que cet élément n'aura pas été envoyé en ligne, le processus de réception MAE restera bloqué grâce à l'opération P sur le sémaphore attente.

Un sémaphore commun à plusieurs processus semcom sert à "réveiller" le processus de formation du message ligne.

S'il s'agit par contre d'une commande valide, elle sera analysée et exécutée (partie interprétation des commandes). Il est impossible d'en donner le déroulement exact.

En tout cas, les commandes à envoyer sont formées dans une zone spéciale dont l'accès est protégé par le sémaphore d'exclusion mutuelle accès zone commandes à envoyer ; le processus de réception MAE positionne le drapeau flag 2 et utilise le sémaphore semcom pour réveiller le processus "formation message ligne".
Nous avons représenté l'algorithme de notre processus sur la figure 34.
L'exclusion mutuelle de la lecture et de l'écriture sur la machine à écrire de pupitre est symbolisée par le sémaphore MAE.

Remarquons que l'attente d'un message doit pouvoir être interrompue par une sortie de message.

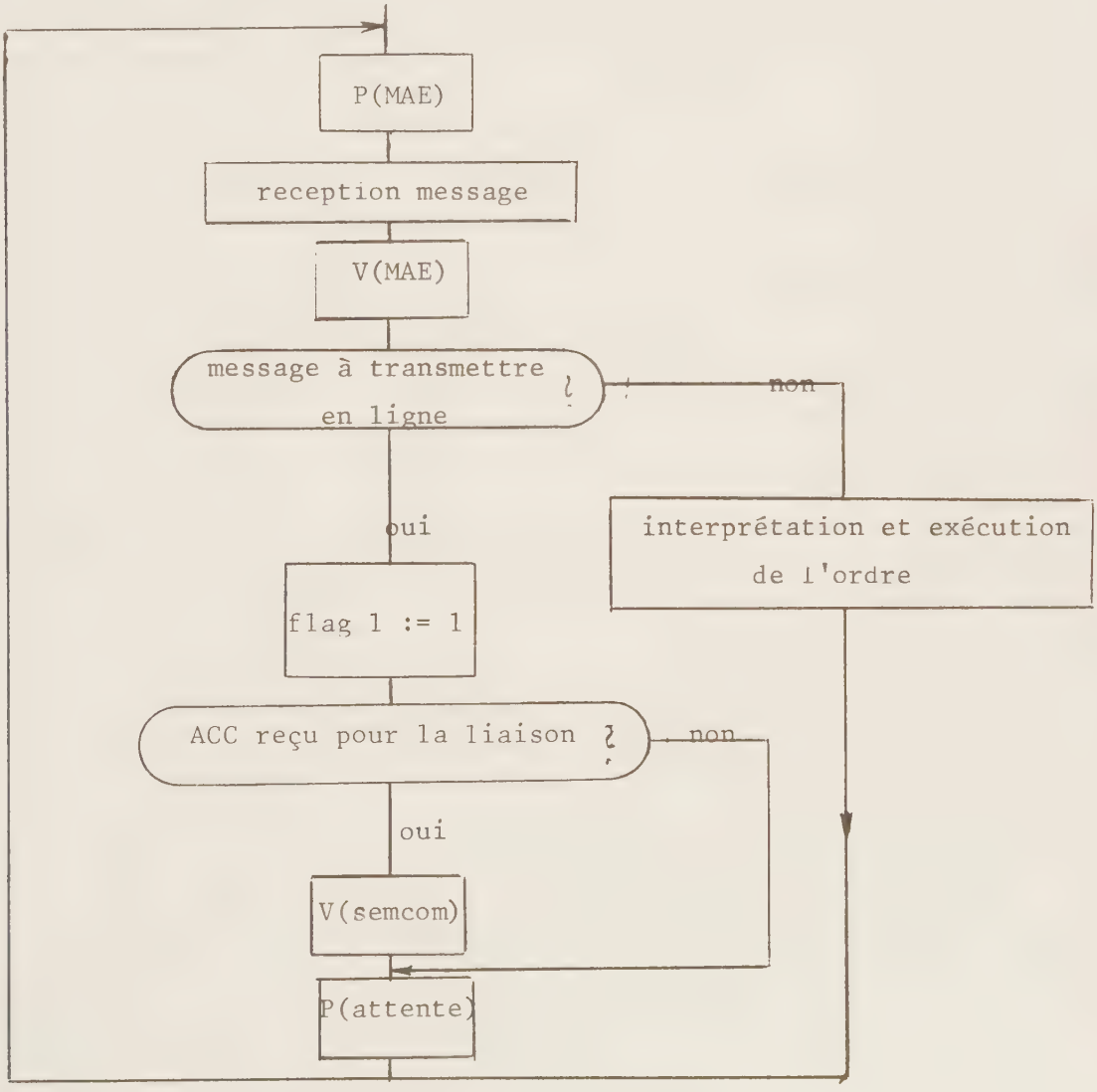


Figure 34 Algorithme du processus réception MAE

4.313 Etablissement des sous-canaux

Les commandes introduites à la MAE peuvent concerner l'établissement d'un sous-canal, sa rupture, les réactions aux anomalies ou l'arrêt général du programme ("on va fermer").

Les renseignements sur les sous-canaux établis ou en cours d'établissement seront consignés dans des tables distinctes suivant qu'il s'agit de sous-canaux SOURCE ou de sous-canaux PUITS DE DONNEES.

Pour la première catégorie de liaisons nous créerons une table des sous-canaux actifs source (TSCAS) ; l'exclusion mutuelle d'accès à cette table sera assurée par le sémaphore accès TSCAS initialisé à la valeur 1.

Chaque entrée dans TSCAS comportera entre autres les informations suivantes :

- numéro du sous-canal
- nom et caractéristiques du fichier lu
- nom du terminal servant de support au fichier
- adresse de début du tampon composant
- longueur de ce tampon

Nous admettons, en effet, que chaque sous-canal disposera de son tampon composant qui sera alloué lors de l'interprétation de la commande d'établissement tapée à la MAE.

Voici l'algorithme d'allocation que nous proposons :
Soit l_1 la longueur maximum d'article du fichier considéré (nous la supposons connue);
Soit l_2 la longueur maximum de la zone de données comprenant des éléments autres que du sous-canal MAE fixée en tenant compte du taux d'erreurs de transmission.

Désignons par ld la place mémoire disponible et par l la longueur base d'allocation.

Nous distinguerons deux cas suivant que l'article du fichier aura ou n'aura pas à être découpé pour sa transmission en ligne :

Cas 1 : $l_1 \leq l_2$ l'article ne sera pas segmenté

Si $ld \geq l_2$ la longueur base 1 correspondra au nombre maximum d'articles dont la somme des longueurs reste inférieure à la longueur maximum de la zone données ce qui s'écrit

$$l = \left[l_2 / l_1 \right] * l_1$$

Si la place disponible ld est inférieure à la longueur maximum de la zone de données tout en restant supérieure à une certaine limite et à la longueur de l'article, nous calculons la longueur base 1 comme correspondant au nombre maximum d'articles la somme des longueurs desquels reste inférieure à la place disponible.

$$\begin{cases} l_1 \leq ld < l_2 \\ \text{et} \\ kl_2 \leq ld \end{cases} \qquad l = \left[ld / l_1 \right] * l_1,$$

kl_2 représente la limite inférieure.

Lorsque la place disponible est inférieure à la limite kl_2, nous considérons que l'allocation est impossible et le sous-canal ne pourra pas être établi.

La longueur effectivement allouée sera supérieure à la longueur base 1 pour tenir compte des informationx auxiliaires (numéro sous-canal, structure, longueur) qui se retrouveront également dans le tampon correspondant On ajoutera à la longueur base une certaine longueur de réserve l_{res} dont la détermination est facile dans le cas d'articles de longueur fixe :

$$l_{res} = 4 + 21/l_1$$

4 octets de description de l'élément et 2 octets longueur par article.

Il est impossible de la prédire pour les articles de longueur variable ; on peut prendre par mesure de précaution :

$$l_{res} = 4 + 4\ 1/l_1$$

Cas 2 : $l_1 > l_2$ l'article devra être segmenté.

Si la place disponible est supérieure à la longueur maximum de l'article la longueur base l sera rendue égale à cette dernière, sionon la demande sera rejetée et le sous-canal ne pourra être établi.

Comme dans le cas 1 on ajoute une longueur de réserve l_{res}.

La figure 35 résume l'algorithme que nous venons de décrire.

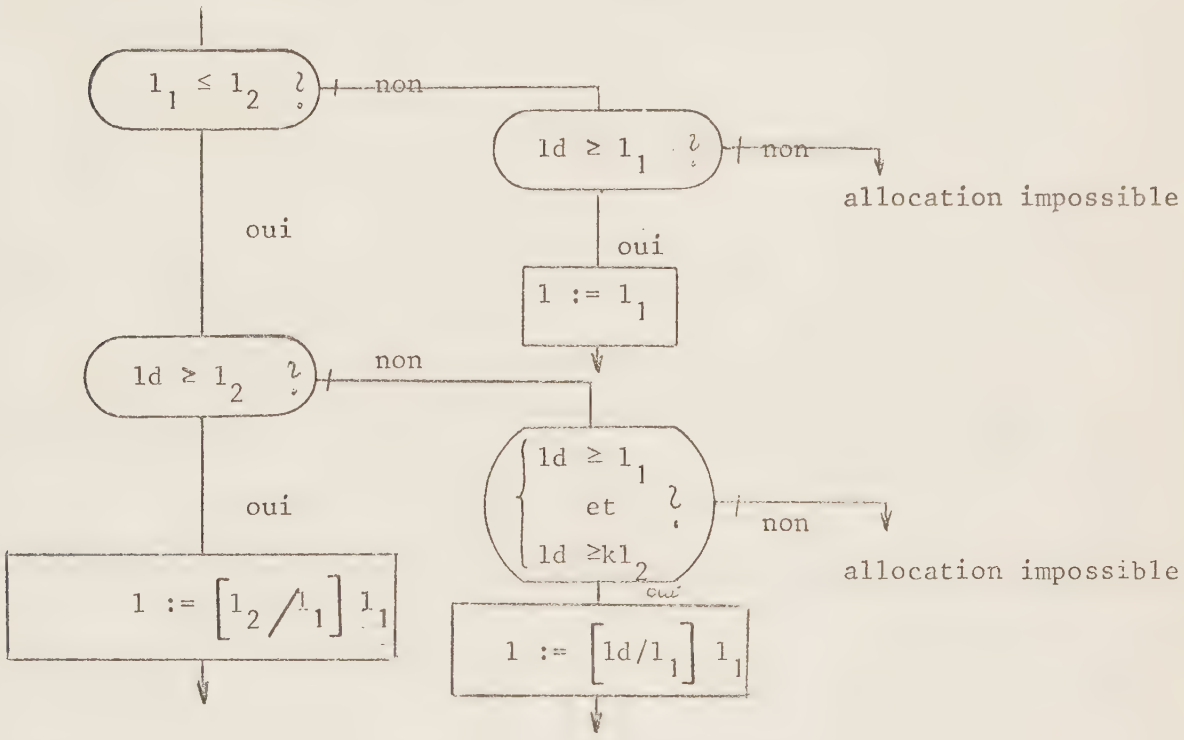


Figure 35 Algorithme d'allocation de tampon composant pour un sous-canal Le même algorithme sera utilisé par le côté PUITS DE DONNEES pour la liaison.

Nous ne voulons pas aborder les problèmes généraux de l'allocation de la mémoire qui ont fait l'objet de nombreux travaux.

Remarquons simplement que l'algorithme tel qu'il vient d'être présenté conduit à la longue à une sous-utilisation de la mémoire par le biais de la fragmentation.

On peut montrer moyennant certaines hypothèses que l'on a plutôt intérêt à allouer la mémoire (et a la libérer) par "paragraphes" contigus de taille fixe. En effet, (la démonstration qui suit est due à M. E. GELENBE [8]) considérons l'état d'équilibre dans lequel toutes les demandes de mémoire sont satisfaites.

Appelons segment une demande de mémoire.

Désignons par No le nombre de segments en mémoire sans paragraphisation

et par N le nombre de segments en mémoire avec paragraphisation

$\rho = \frac{N}{No}$ est une mesure d'efficacité de la paragraphisation.

Soient

- μ la taille de la mémoire
- π la taille du paragraphe
- q_i la probabilité qu'un trou soit de taille i
- p_i la probabilité qu'un plein soit de taille i ou, ce qui revient au même sous équilibre, que l'on doive allouer i
- z_j la probabilité qu'il y ait j trous

La probabilité p qu'une demande ne trouve pas un trou de sa taille s'écrit alors

$$p = \sum_{i=1}^{\infty} p_i \sum_{j=0}^{\infty} z_j \, (1 - q_i)^j \qquad (1)$$

Si un segment est de taille x on alloue $\left\lceil \frac{x}{\pi} \right\rceil$ paragraphes ($\lceil \ \rceil$ = entier immédiatement supérieur ou égal)

Appelons r la taille moyenne d'un plein et soit λ la taille moyenne d'un segment.

$$r = E\left\{\left\lceil \frac{x}{\pi} \right\rceil \pi\right\} = \lambda + \frac{\pi}{2} + \gamma \qquad (2)$$

pour un grand nombre de distributions de probabilité (uniforme, Erlang) $\gamma = o$ et donc $r = \lambda + \frac{\pi}{2}$ (3)

Gelenbe montre [7] qu'en équilibre on peut écrire $\mu = N\left(1 + \frac{p}{2}\right)$,

μ étant la taille de la mémoire.

On a alors

$$N = \frac{\mu}{r} \quad \frac{1}{1+\frac{p}{2}}$$

et

$$No = \frac{\mu}{ro} \quad \frac{1}{1+\frac{po}{2}} \qquad ,ro = E\{x\} = \lambda \ ; \ po \simeq 1 \text{ (confirmé par les expériences)}$$

$$\rho = \frac{N}{No} = \frac{ro}{r} \quad \frac{1 + \frac{po}{2}}{1 + \frac{p}{2}} = \frac{2+po}{2+p} \quad \frac{ro}{r}$$

d'où $$\rho = \frac{\lambda}{r} \quad \frac{3}{2+p} \qquad (5)$$

Nous allons montrer maintenant que ρ peut être supérieur à l'unité dans certaines conditions et que donc la mémoire sera mieux utilisée si elle est divisée en paragraphes.

$$p = \sum_{i=1}^{\infty} p_i \sum_{j=o}^{\infty} z_j \ (1-q_i)^j \simeq \sum_{i=1}^{\infty} p_i \sum_{j=o}^{\infty} z_j (1-jq_i) \qquad (6)$$

en admettant que $q_i << 1$

(6) s'écrit alors

$$\sum_{i=1}^{\infty} p_i \sum_{j=o}^{\infty} z_j - \sum_{i=1}^{\infty} p_i q_i \sum_{j=o}^{\infty} j z_i$$

ce qui donne

$$p = 1 - M \sum_{i=1}^{\infty} p_i q_i \qquad (7)$$

$M = \sum_{j=o}^{\infty} j z_j$, M est donc le nombre moyen de trous

Supposons que p_i suive une distribution uniforme ; $p_i = \frac{\pi}{2\lambda}$ (on suppose que $\pi << \mu$ c'est-à-dire que la taille d'un paragraphe est beaucoup plus petite que celle de la mémoire).

Posons $\varepsilon = \frac{\pi}{2\lambda}$ (7) devient $p = 1 - M\varepsilon$, quelle que soit la distribution des q_i.

D'après (5) et (3) $\rho = \frac{\lambda}{\lambda + \frac{\pi}{2}} \quad \frac{3}{2+p} = \frac{1}{1+\varepsilon} \quad \frac{3}{2+P}$

En tenant compte de (7) on obtient

$$\rho = \frac{1}{1+\varepsilon} \quad \frac{3}{3 - M\varepsilon} \simeq \quad \frac{1 - \varepsilon}{1 - \varepsilon \frac{M}{3}}$$

On voit que si M est assez grand (ce qui est le cas habituellement) $\rho > 1$.

Pour la deuxième catégorie de liaisons nous créerons une table des sous-canaux actifs puits (TSCAP) analogue à la table TSCAS.
Le sémaphore accès TSCAP initialisé à la valeur 1 assurera l'exclusion mutuelle d'accès à cette table.
Comme nous l'avions mentionné lors de la définition de la commande d'établissement de sous-canal DEM 2, nous créerons une table des sous canaux PUITS en cours d'établissement (figure 36).

n° provisoire sous-canal	device	Réservation mémoire	n° sous-canal de DEM 1

si sous-canal "volé" par DEM 1

Figure 36

Si le sous-canal a été "volé" par DEM 1 avant la réception de ADEM2, la liaison établie par ADEM 2 sera rompue par l'envoi d'une commande RPL. Dans le cas contraire un ACC sera émis par le PUITS DE DONNEES ; la SOURCE ne génère pas de ACC pour les liaisons demandées par DEM 2, car cela pourrait conduire à envoyer des données pour un sous-canal que le PUITS vient d'annuler si la commande RPL ne parvenait pas à temps.

4.32 Fonction R

La partie R sera conçue comme un ensemble de processus coopérants qui assurent la réception des messages sur la ligne, leur analyse, l'interprétation et exécution des ordres que comporte la zone de commandes (si elle est présente), la reconstitution des articles dans les tampons composants à partir des éléments présents dans la zone de données du message, et l'écriture de ces articles dans les fichiers correspondants aux sous-canaux concernés.

Nous choisissons l'organisation à un buffer de réception ligne pour les messages transmis et un tampon composant par sous-canal PUITS.

Le sous-canal de communication entre les machines à écrire de pupitre sera traité de façon analogue aux autres sous-canaux pour ce qui est de la transmission en ligne,ce qui veut dire que l'on enverra des accusés de réception.

4.321 Processus de réception et d'analyse des messages ligne.

Commençons par le processus de réception des messages ligne dont l'algorithme est représenté sur la figure 37.

On effectue d'abord un polling sur la ligne c'est-à-dire on se met en attente d'un message émis par l'ordinateur correspondant.
Une fois ce message reçu,on en effectue l'analyse qui a pour but de déterminer les zones présentes et à l'intérieur de la zone de données (si celle-ci fait partie du message, bien sûr) de délimiter les éléments,pour en communiquer ensuite les adresses aux différents processus de sortie réveillés par l'opération V sur le sémaphore processus i correspondant.

Les processus de sortie transféreront en premier lieu les éléments dans leurs tampons composants. La fin de ce transfert est attendue par une suite d'opérations P dont le nombre est égal au nombre de processus de sortie réveillés.

Il nous a semblé commode d'introduire une opération sur un sémaphore que nous appelons $P_n(s)$. $n \geqslant 1$, entier
Elle a pour effet de décrémenter la valeur du sémaphore S de n dès que le résultat serait non négatif. Elle est équivalente à une suite de n opérations P sur s.
On peut définir de façon analogue l'opération $V_n(s)$.

Entre temps on interprète et exécute les ordres de la zone de commandes. Ceci peut entraîner la consultation et des modifications des tables TSCAS et TSCAP. L'exclusion mutuelle des processus dans la manipulation de ces tables est assurée par les sémaphores accès TSCAS et accès TSCAP.

Si des accusés de réception (ACC ou NACC) ont été trouvés dans la zone de commandes, ils sont placés dans une table spéciale. Les commandes que le processus voudra faire transmettre en ligne, devront être formées dans la zone des commandes à envoyer dont l'accès est protégé par le sémaphore accès zone commandes à envoyer ; on positionne le drapeau flag 2 s'il était à 0 et on effectue l'opération V sur semcom,

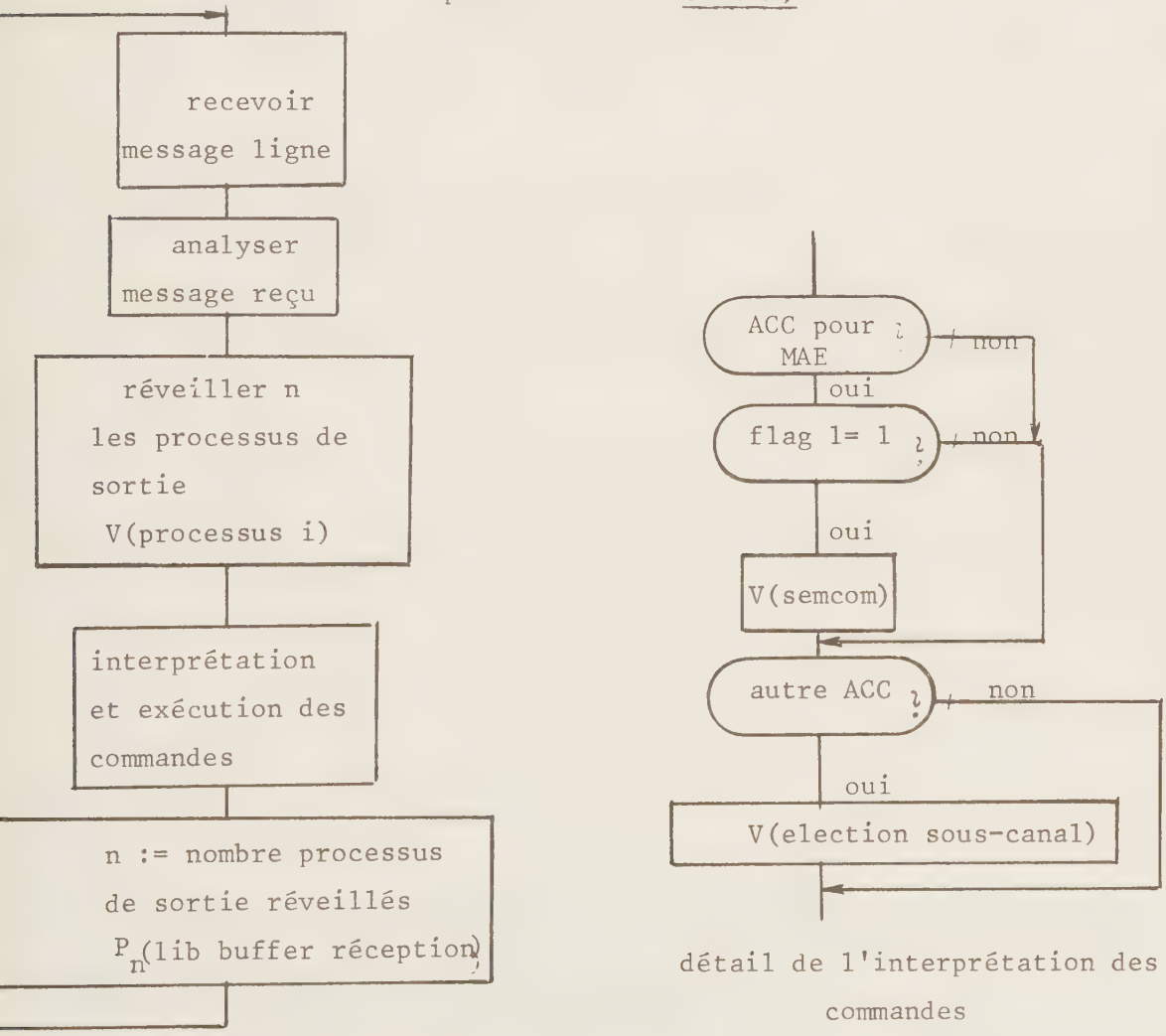


Figure 37. Algorithme du processus de réception des messages en ligne.

Lorsque l'opération P_n sur le sémaphore lib buffer réception a pu être terminée, on est sûr que le tampon de réception ligne peut être écrasé et la réception d'un nouveau message est alors possible.

Voici en résumé les sémaphores utilisés avec leurs valeurs d'initialisation :

- lib buffer réception : 0,
- processus i : 0,
- semcom : 0,
- accès TSCAS : 1,
- accès TSCAP : 1,
- accès zone commandes à envoyer : 1,
- accès table sous-canaux en cours d'établissement : 1,
- élection sous-canal : 0,

4.322 Processus de sortie dans le fichier

Les processus de sortie sont synchronisés avec celui de réception et analyse du message ligne par l'intermédiaire des sémaphores processus . On leur transmet au "réveil" l'adresse de début de l'élément dans le buffer de réception.

Nous rappelons sur la figure 38 la forme d'un élément destiné à un sous-canal (autre que celui de communication entre les MAE)

Le transfert de l'élément du buffer ligne dans le tampon composant dépend de l'octet structure dont les valeurs possibles sont :

- article complet
- plusieurs articles complets
- début article
- partie article
- fin article

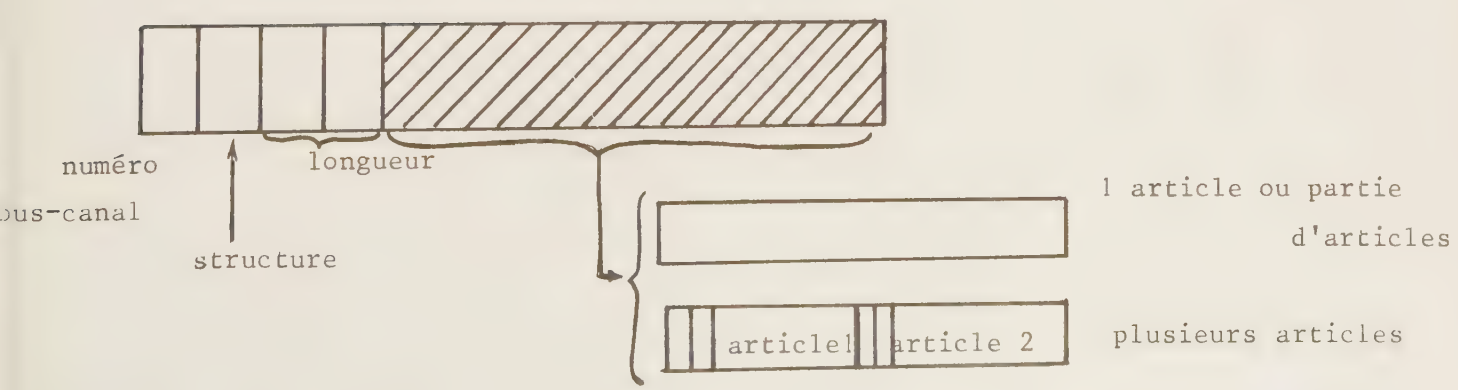


Figure 38

Dans les 3 premiers cas on transfère tout l'élément à part l'octet "numéro sous-canal".
Lorsqu'il s'agit d'une suite d'article (partie ou fin d'article), on recopie juste la partie d'article en la plaçant dans le tampon composant comme le montre la figure 39.

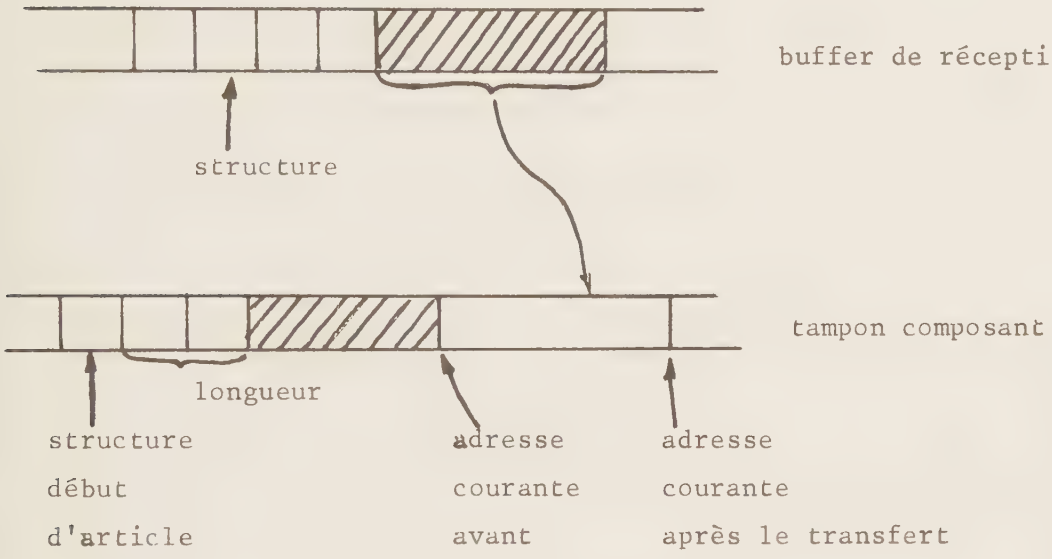


Figure 39

Tout processus de sortie doit tenir à jour l'adresse courante dans le tampon ; elle pourrait être consignée dans la table TSCAP.
Lors de chaque transfert dans le tampon composant, il est indispensable de mettre à jour la longueur des données s'y trouvant ; on le fait simplement en ajoutant les contenus des 2 zones longueurs (ancienne et celle qui vient d'être transférée).

Quand l'article est complet nous pouvons vider le tampon par une écriture dans le fichier lié au sous-canal.
L'algorithme du processus se trouve sur la figure 40.
Nous rassemblons ci-dessous les sémaphores utilisés par le processus avec leurs valeurs d'initialisation :

. processus i : 0 ;
. lib buffer réception : 0,
. semcom : 0,

. accès TSCAP : 1,

. accès zone commandes à envoyer : 1.

Le drapeau flag 2 est positionné signalant la présence de commandes dans la zone des commandes à envoyer.

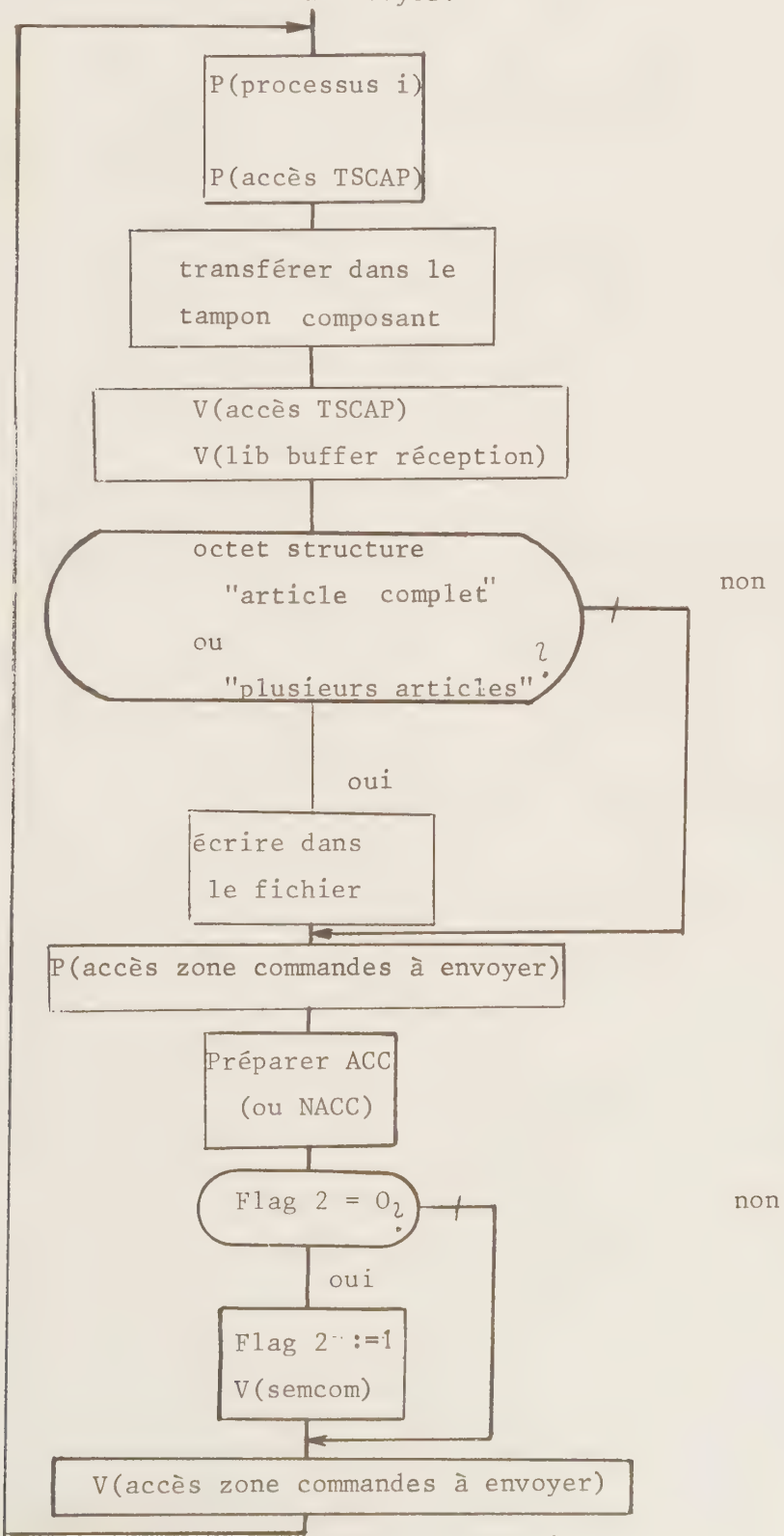
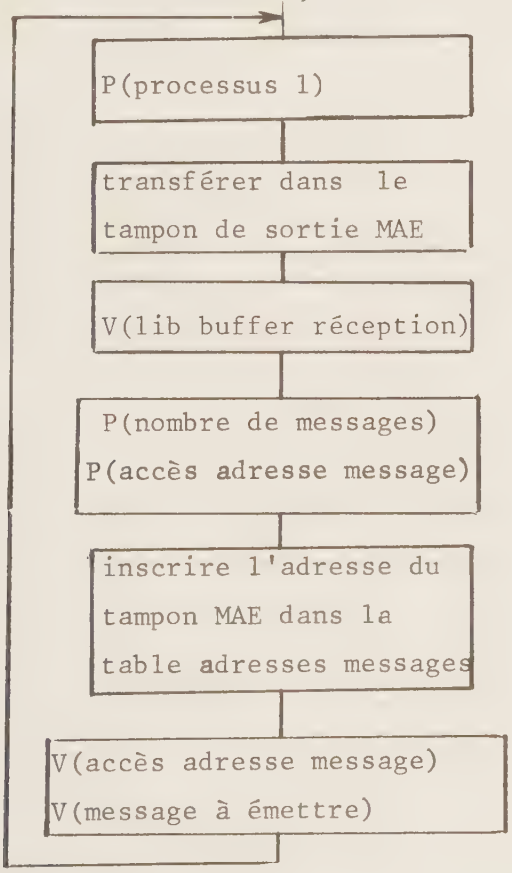


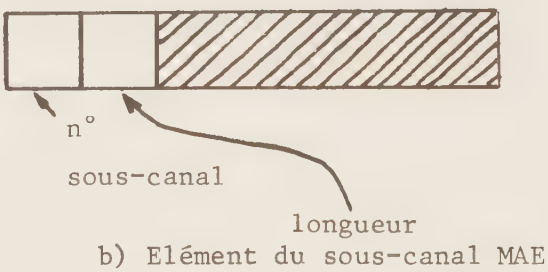
Figure 40 Algorithme d'un processus de sortie

Il y a un processus de sortie un peu particulier ; c'est celui lié au sous-canal de communication entre les opérateurs.
Ses seules fonctions consistent à transférer l'élément à partir de l'octet longueur dans le tampon de sortie MAE, et à placer l'adresse de ce tampon dans la table du processus de sortie sur la MAE (voyez figure 41).

Rappelons que tout processus voulant sortir un message à la MAE locale doit se synchroniser à l'aide des sémaphores nombre de messages et accès adresse message.

Nous ne faisons pas figurer cette séquence sur les algorithmes ; elle est sous-entendue à chaque réaction aux anomalies, chaque décision laissée à l'utilisateur, etc.





b) Elément du sous-canal MAE

Figure 41 a) Algorithme du processus de sortie du sous-canal MAE

4.33 Fonction E

La partie E, elle aussi,sera constituée d'un ensemble de processus coopérants. Ces processus se chargeront de la lecture des données dans les fichiers, de la formation des éléments et du message ligne de même que de l'émission de ce dernier en ligne.

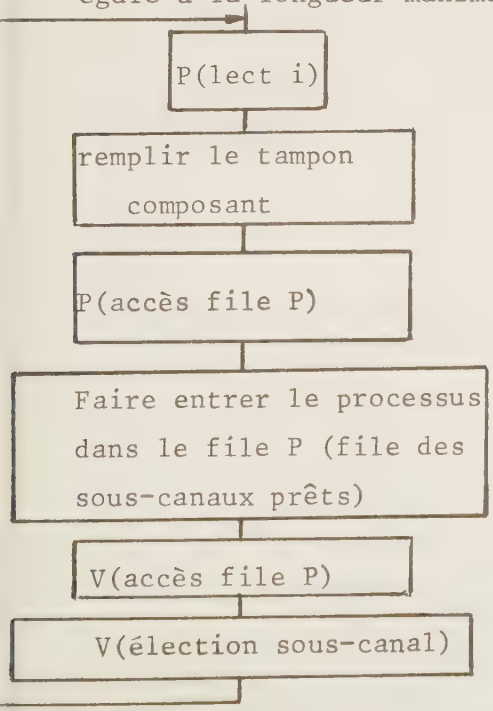
Nous choisissons l'organisation à autant de tampons composant que de sous-canaux SOURCE DE DONNEES, l'émission en ligne ayant lieu avec chaînage des zones de commandes et de données (propriété 1 de notre système).

4.331 Processus de garnissage du tampon composant

Considérons d'abord les processus qui assurent le remplissage des tampons composant par lecture dans les fichiers liés aux sous-canaux (algorithme figure 42).

La synchronisation du processus avec d'autres se fait par le truchement du sémaphore lect i ; la lecture des articles du fichier entraîne une consultation de la table TSCAS l'exclusion mutuelle d'accès à laquelle a lieu grâce au sémaphore accès TSCAS.

Signalons la difficulté qui apparaît lors de la lecture des articles de longueur variable ; pour être certain de ne pas déborder du tampon,il faut s'astreindre à ne demander la lecture de l'article suivant que si la place encore disponible dans le tampon composant est au moins égale à la longueur maximum d'article plus deux octets (longueur de l'article).



Une fois le tampon composant rempli,le sous-canal est considéré comme prêt et sera placé dans la file d'attente des sous-canaux prêts - file P.
L'accès en est protégé par l'intermédiaire du sémaphore accès file P.
En effectuant une opération V sur le sémaphore élection sous-canal,nous avertissons le processus chargé de choisir le (ou les) sous-canal(aux) dont l'élément fera partie du prochain message.

Figure 42 Algorithme du processus de garnissage du tampon composant

Voici les valeurs d'initialisation des sémaphores utilisés par le processus :

. lect i : o,

. accès file P : 1,

. accès TSCAS : 1,

. élection sous-canal : 0.

4.332 File des sous-canaux SOURCES DE DONNEES prêts (file P)

Tout sous-canal prêt c'est-à-dire dont l'élément est déjà formé dans le tampon composant, à l'exception du sous-canal de communication entre les MAE de pupitre, rejoint la file d'attente des sous-canaux prêts - file P. L'entrée dans la file se fait toujours en queue.

Pour déterminer le premier élément de la file à en sortir, nous la parcourons à partir du début en choisissant le premier sous-canal pour lequel un accusé de réception positif (ACC) a été reçu ou généré (voyez figure 43).

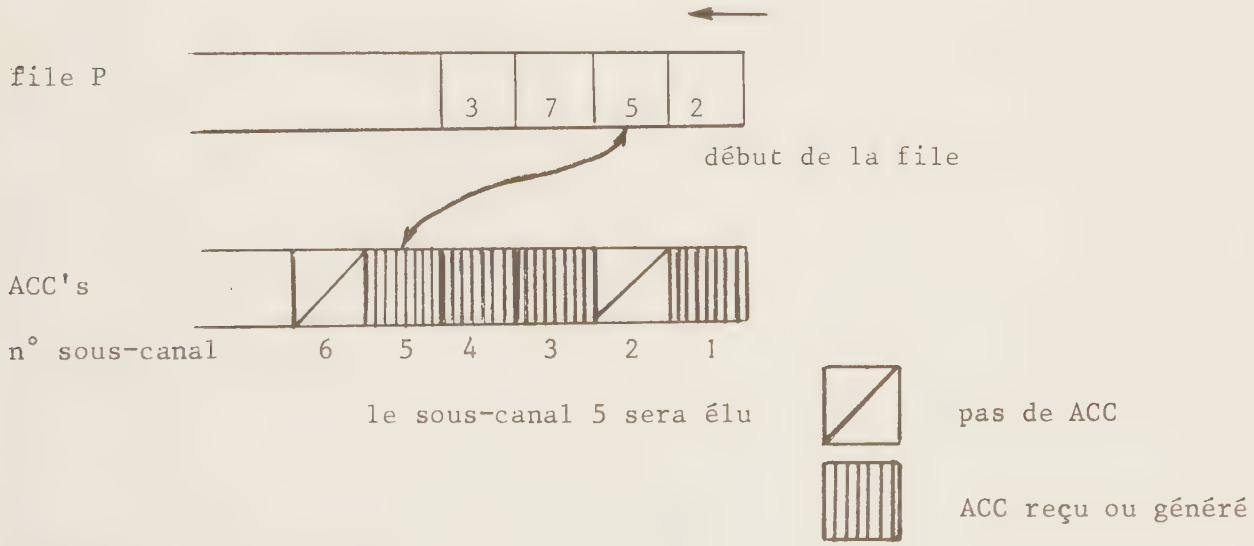


Figure 43 Sortie de la file P.

Le sous-canal ainsi élu quitte la file P sauf s'il s'agit d'un article découpé pour les besoins de la transmission en ligne, auquel cas le sous-canal garde sa place dans la file P tant qu'il reste une partie de l'article dans le tampon composant*.

Un processus spécial de gestion de la file P représenté sur figure 44 réalise l'algorithme de sortie.

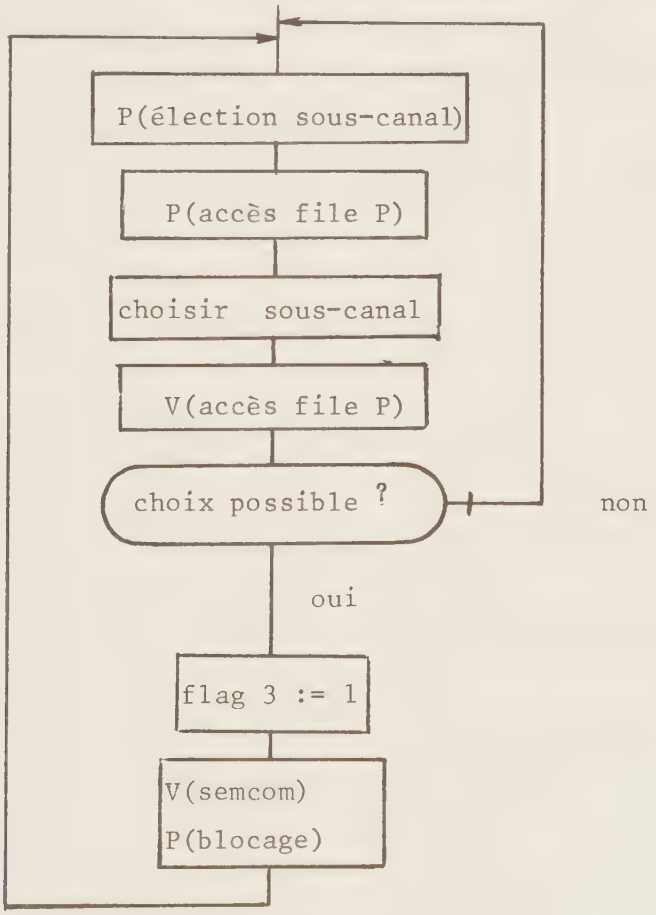


figure 44 Processus de gestion de la file P.

*On assure de cette façon une sorte de priorité à ce sous-canal ; on pourrait concevoir de remettre le sous-canal en queue...

Le processus ne choisit qu'un seul sous-canal ; il range dans une table spéciale les renseignements concernant celui-ci.

Si la longueur de l'élément n'impose pas une fragmentation de l'article,le sous-canal est enlevé de la file P ; dans tous les cas l'ACC correspondant est effacé.

On positionne le drapeau flag 3 et une opération V est effectuée sur le sémaphore semcom pour "réveiller" le processus de formation du message ligne qui devra effectuer une opération V sur le sémaphore blocage.

Les sémaphores utilisés sont :

. élection sous-canal initialisé à la valeur : 0,

. semcom : 0,

. blocage : 0,

. accès file P initialisé à la valeur : 1.

4.333 Processus de formation du message ligne

Le processus de formation du message ligne commence par attendre l'arrivée d'un événement au moins parmi les 3 suivants :

- il y a des commandes à envoyer
- il y a un élément pour le sous-canal MAE et un ACC a été reçu pour ce sous-canal
- il y a un élément pour un autre sous-canal qui a été choisi par le processus de gestion de la file P.

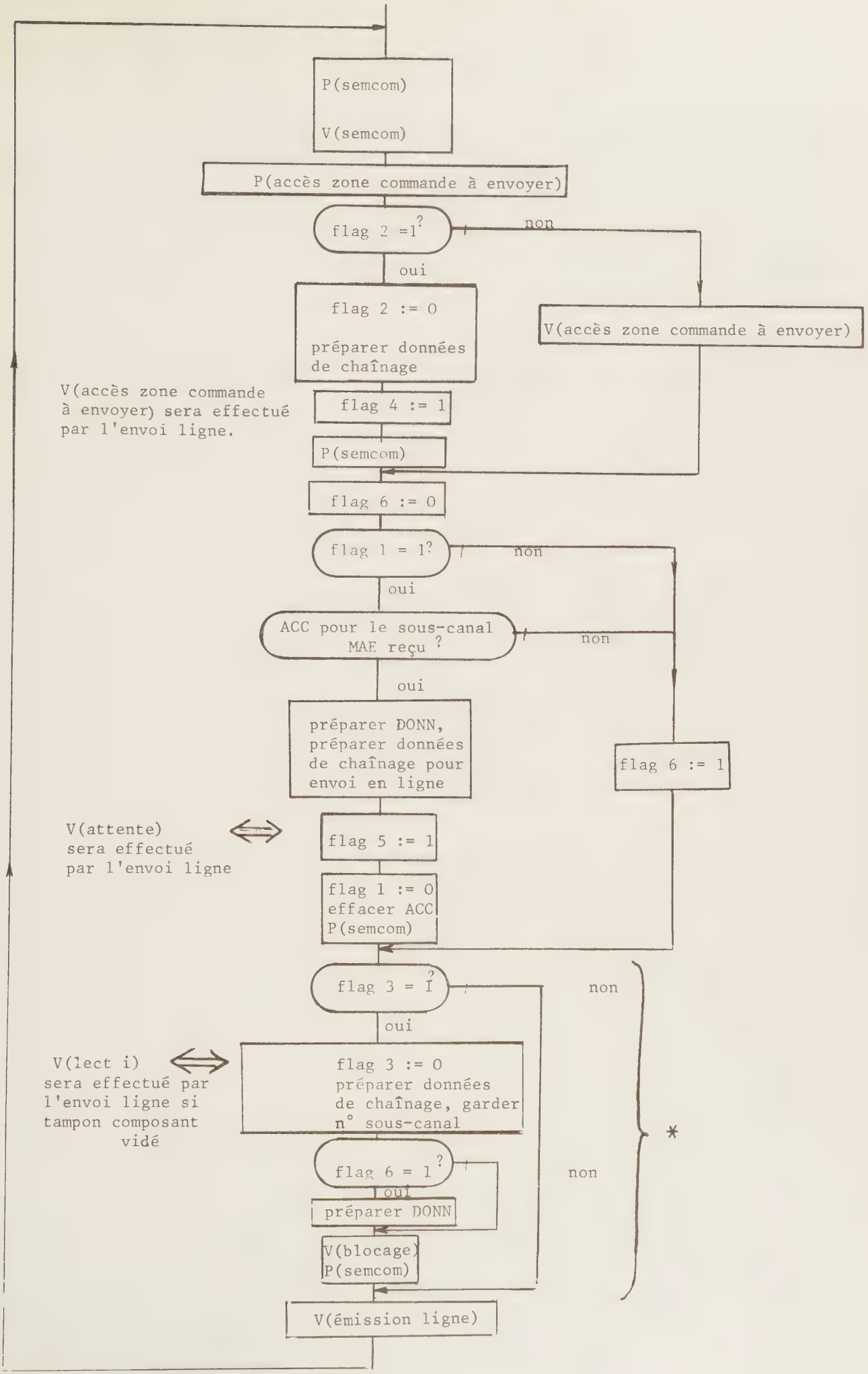


Figure 45 Algorithme du processus de formation du message de transmission en ligne.

Cette attente est réalisée grâce au sémaphore commun semcom . Une fois "réveillé",le processus commence par tester le drapeau flag 2 qui lui permet de savoir s'il y a des commandes à envoyer. Si tel est le cas,on prépare les données de chaînage pour l'envoi en ligne. En mettant le drapeau flag 4 à 1 on marque qu'une opération V sur le sémaphore accès zone commandes à envoyer devra être effectuée.

Le processus continue en testant la valeur du drapeau flag 1. Lorsqu'un élément pour le sous-canal de communication entre les machines à écrire de pupitre attend dans son tampon et qu'un accusé de réception positif - ACC - a été reçu pour ce même sous-canal, on chaîne cet élément en le faisant précéder de l'identifieur de la zone de données (DONN) que l'on peut placer à la suite des commandes à envoyer,ce qui sera marqué par le positionnement du drapeau flag 6.

L'ACC pour le sous-canal MAE est effacé, flag 1 est remis à 0 ; la valeur 1 du drapeau flag 5 permettra de savoir au processus d'émission sur la ligne qu'une opération V doit être faite sur le sémaphore attente. On inspecte ensuite la valeur de flag 3. Si le drapeau est à 1 on chaîne l'élément du sous-canal élu par le processus de gestion de la file P, lequel est ensuite débloqué grâce à l'opération V sur le sémaphore blocage.

Le processus d'émission du message ligne est réveillé par l'intermédiaire du sémaphore émission ligne.

Si l'on veut réaliser un multiplexage dans l'espace (plusieurs sous-canaux dans le même message), il faut répéter la séquence marquée par un astérisque sur la figure 45.

Nous rassemblons ci-dessous les sémaphores utilisés par le processus avec leurs valeurs d'initialisation.

. semcom : 0,
. accès zone commandes à envoyer : 1,
. blocage : 0,
. émission ligne : 0.

4.334 Processus d'émission du message en ligne

Il nous reste à voir le processus chargé de l'émission du message sur la ligne. Les données de chaînage pour la commande canal ont été préparées par le processus de formation du message. Il suffit maintenant d'émettre, d'autoriser l'accès à la zone des commandes à envoyer, et de "réveiller" le processus de réception MAE et les processus de garnissage du tampon composant.

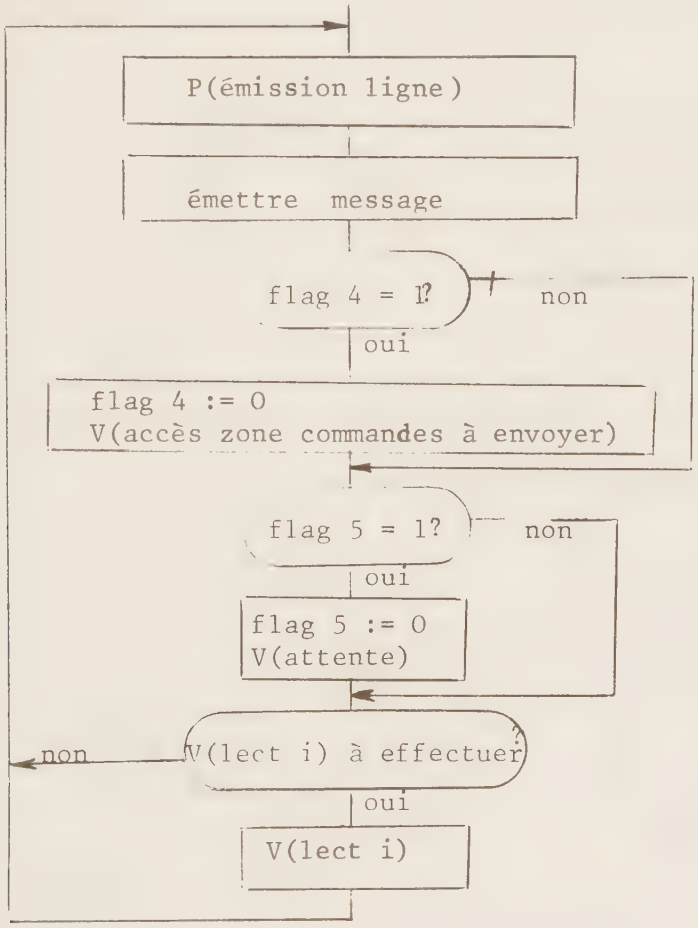


Figure 46 Algorithme du processus d'émission en ligne.

Voici les sémaphores utilisés avec leurs valeurs d'initialisation.

. émission ligne : 0,
. accès zone commandes à envoyer : 1,
. attente : 0,
. lect i : 0.

Les traits essentiels de la structure du programme sont indiqués ci-dessous (figure 47).

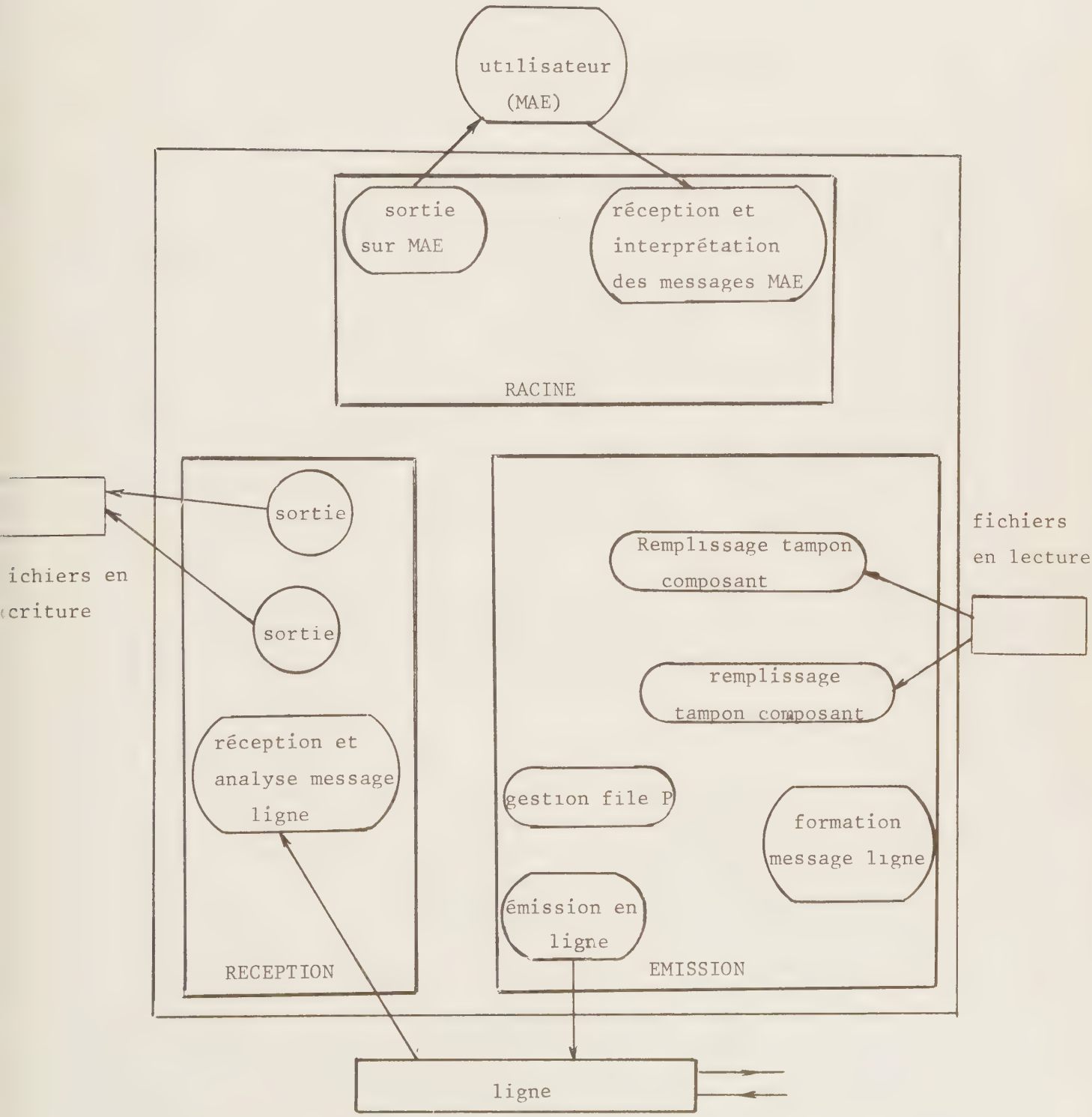


Figure 47

5. Extension de la procédure d'exploitation à un réseau de transmission

Nous avons envisagé jusqu'à présent une liaison bi-point (entre 2 ordinateurs seulement). On peut concevoir une mise en commun des moyens de calcul au niveau national ou même international réalisée par une interconnexion des ordinateurs en un réseau.

Au prix de quelques modifications la procédure d'exploitation définie dans 2 peut être étendue à ce cas-là.

Un tel réseau peut avoir des organisations différentes, il peut être très centralisé ou, au contraire, présenter une grande dispersion.

Il est licite de considérer un réseau d'ordinateurs comme un ensemble de liaisons bi-point quelle que soit la façon réelle dont ces liaisons sont établies et donc de garder la notion de sous-canal.

Chaque sous-canal sera défini désormais par 4 paramètres qui sont, en toute généralité, les terminaux mis en correspondance et les ordinateurs auxquels ces terminaux sont connectés.

Nous convenons que chaque message s'adresse à un seul des correspondants c'est-à-dire que l'ensemble des commandes et des données contenues dans un message est destiné aux sous-canaux d'un seul ordinateur.

Il apparaît que la principale modification à introduire consiste à ajouter à chaque message une indication permettant de connaître l'ordinateur qui l'a émis (figure 48).

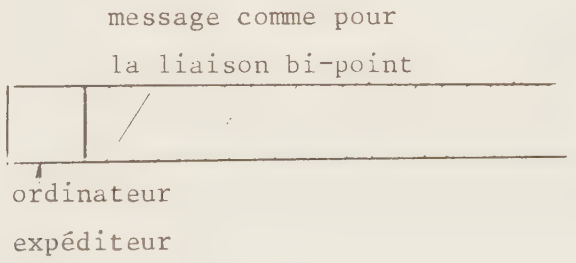


Figure 48.

Bien entendu, des changements seront également indispensables dans certaines fonctions (processus), c'est ainsi qu'il faudra, par exemple, indiquer maintenant auquel des ordinateurs-correspondants s'adresse un message introduit à la console de l'opérateur.

Pour ce qui est de la gestion des sous-canaux SOURCES DE DONNEES elle pourrait rester la même : on constituerait une file d'attente des sous-canaux prêts (c'est-à-dire pour lesquels un élément est déjà formé).

A chaque ordinateur avec lequel une (ou plusieurs) liaison est ou doit être établie correspondrait une zone des commandes à envoyer.

Pour attribuer une priorité à la communication entre les opérateurs des différentes calculatrices,nous sommes obligés de placer le sous-canal correspondant en tête de la file P.

Une autre solution, préférable peut-être, consiste à créer une file des sous-canaux prêts par ordinateur PUITS DE DONNEES - correspondant et une file d'attente circularie C (voir figure 49) des ordinateurs prêts c'est-à-dire dont la file P n'est pas vide.

La file C permettrait de choisir quelle liaison bi-point (entre ordinateurs) on doit prendre en considération.

Cette façon de procéder qui revient à introduire une certaine hiérarchie dans la gestion des liaisons,donnerait la possibilité de regrouper les sous-canaux d'un même ordinateur et favoriserait par cela même le multiplexage dans l'espace.

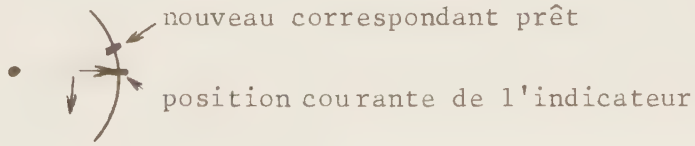


Figure 49. La file C d'attente des ordinateurs prêts.

Une gestion analogue (à 2 niveaux) paraît recommandable au cas où une priorité serait attachée à chaque ordinateur correspondant.

6. Exemple d'application à l'ordinateur CII IRIS 50

6.1 Objectifs

Nous avons voulu programmer une application simple de la procédure d'exploitation et d'une partie des algorithmes définis précédemment afin de tester leurs rationalité et facilité d'emploi.

Nous avons choisi comme exemple une conversion cartes-disque et disque-imprimante qui pourrait bien représenter la coopération de deux centres de calculs : les cartes du job ayant été lues dans le centre A leur image est stockée sur disque dans le centre B ; après l'exécution les résultats sont transmis vers l'imprimante de A. Notre but était d'écrire un programme relativement simple et structuré plutôt que de chercher à tout prix à optimiser ses performances.

6.2 Choix de l'ordinateur

6.21 Généralités

Deux sortes de raisons ont présidé au choix de l'ordinateur IRIS 50 de la CII pour cette application :

- la première, d'ordre "historique", est que nous avons commencé notre étude à la Compagnie Internationale pour l'Informatique sur cette machine,
- la deuxième tient à la structure du calculateur dont un très grand avantage est d'avoir la mémoire adressable par octets et d'offrir des instructions câblées de traitement de chaînes de caractères, ce qui permet de les manier avec beaucoup de facilité.

D'autre part, IRIS 50 est muni de systèmes de Gestion de Fichier (SGF) et de Gestion de Transmission (SGT) qui devaient, a priori, rendre plus aisé l'accès aux fichiers et à la ligne de transmission. Un examen un peu plus approfondi du système existant permet cependant de réaliser que celui-ci n'est pas bien adapté à nos besoins.

En effet, premièrement, l'allocation de la mémoire est statique, la réservation étant faite à l'assemblage, deuxièmement, il est difficile sinon impossible de se référer de façon dynamique à un fichier de l'intérieur d'un programme puisque normalement le fichier doit être décrit à l'aide de macro-instructions du SGF à l'assemblage,

cette description pouvant être complétée ou partiellement modifiée au moyen de cartes de commande du moniteur.

Cette difficulté provient, entre autres, de ce que notre programme n'est pas un programme utilisateur ordinaire ; il devrait en fait faire partie du système d'exploitation.

6.22 Système de Gestion de Fichiers

Le Système de Gestion de Fichiers (SGF) est organisé en deux niveaux :

- de l'enregistrement physique, dit niveau physique (SGF physique)
- de l'article logique, dit niveau logique (SGF logique).

A chacun de ces deux niveaux correspond un jeu de macro-instructions spécialisées disponibles dans le langage de l'Assembleur.

On distingue deux types d'organisation des fichiers sur le support :

- standard : le fichier est implanté, identifié et répertorié suivant les normes du SGF,
- non standard : le fichier est implanté selon les normes particulières ignorées du SGF.

Un fichier organisé, c'est-à-dire muni de la structure d'articles, ne peut être géré, en tant que tel, qu'en SGF logique. Néanmoins, tout fichier d'organisation "article" peut toujours être considéré comme "non défini" et donc exploité en SGF physique.

Pour les fichiers "définis" trois types d'organisation logique, adaptés aux modes d'accès classiques, sont admis : séquentiel, direct, mixte (séquentiel et direct).

L'accès direct n'est envisagé que sur support adressable.

En ce qui concerne la structure de l'information on distingue trois formats standard d'articles :

format fixe : tous les articles sont de même longueur,

format variable : la longueur de chaque article peut varier,

format non défini : l'article est constitué par l'enregistrement physique.

Sur support magnétique les articles peuvent être groupés (bloqués) ce qui veut dire que les enregistrements physiques contiennent plusieurs articles.

Un fichier d'organisation non définie ne peut être géré qu'en SGF physique.

L'unité d'accès est alors l'enregistrement physique, tout support est admis.

En support non adressable, on autorise l'accès à l'enregistrement physique suivant ou précédent si l'élément externe est doté de lecture arrière.

En support adressable on peut accéder à l'enregistrement physique d'adresse donnée qui est toujours une adresse relative par rapport au début du fichier. Des zones de mémoire centrale doivent être affectées aux informations à transférer vers ou depuis les organes externes. La gestion de ces zones tampon désigne leur vidage (écriture) et leur garnissage (lecture) lors du traitement des fichiers.

Les opérations d'entrée/sortie sont commandées et contrôlées par les macro-instructions EXUP (Execute EXchange Unit Program) et WAIT du moniteur d'entrée/sortie.

Ces macro-instructions se réfèrent à un ensemble Bloc de Commande et Programme d'Unité d'Echange, appelé Bloc de gestion directive (BGD).

La gestion des zones tampon est dite automatique lorsque les associations zone tampon - BGD - EXUP/WAIT et l'exécution des instructions EXUP et WAIT sont entièrement assurées par le SGF.

Dans le cas contraire, la gestion est dite programmée.

La fonction de blocage et déblocage d'articles du SGF implique la gestion automatique des tampons. On comprend dans ces conditions que cette dernière soit la seule possible pour le traitement des fichiers sélectifs et séquentiels indexés. La gestion programmée est la seule admise pour le traitement des fichiers non définis. Les deux modes de gestion peuvent être utilisés en fichier séquentiel.

L'unité de transfert entre bande et mémoire est le bloc qui constitue également l'enregistrement physique ; sur disque l'enregistrement physique est le bloc de longueur égale à un multiple du nombre de secteurs par cylindre. Lorsque la quantité d'information utile par bloc ne le remplit pas entièrement,on transfère seulement la partie utile, appelée "page". L'unité de transfert entre disque et la mémoire est donc la page de dimension inférieure ou égale à celle du bloc.

Remarquons tout de suite que nous devons nous interdire l'emploi de la gestion automatique si nous voulons éviter un gaspillage de place mémoire et de temps.

En effet, puisque l'élément est formé dans le tampon composant, il est rationnel d'effectuer la lecture directement dans ce buffer sans passer par une zone tampon intermédiaire.

Il en est de même pour l'écriture dans le fichier.

D'autre part, en gestion automatique, l'exécution d'une macro-instruction d'accès comporte obligatoirement l'instruction d'attente WAIT ce qui exclut toute simultanéité des entrées/sorties et des traitements autre que celle prévue dans le Système de Gestion de Fichiers lui-même.

Or, comme nous voulons assurer plusieurs conversions simultanées et dans les deux sens, nous avons intérêt à utiliser au mieux les temps d'attente liés aux entrées/sorties pour les traitements.

Aussi faudra-t-il exploiter les fichiers en organisation non définie (niveau physique) ou, pour les fichiers séquentiels, recourir à la gestion programmée des tampons.

Le SGF met à la disposition des utilisateurs des macro-instructions d'accès : organisation du fichier non définie :

READ lecture d'un enregistrement physique

RREV lecture arrière (bande) d'un enregistrement physique

WRITE écriture d'un enregistrement physique

CH_B^W analyse de la fin d'exécution d'une macro-instruction précédente et orientation de l'exécution du programme en fonction du résultat de cette analyse.

Organisation séquentielle en gestion programmée (assurée par l'utilisateur) :

GBF lecture bloc/ page suivant

GBR lecture bloc/page précédent

PBL écriture bloc/page suivant

RBL remplacement du bloc lu

CH_B^W analyse de GBF/GBR/PBL précédent

6.23 Système de Gestion de Transmission

Le Système de Gestion de Transmissions fournit un certain nombre de macro-instructions,permettant premièrement de décrire le réseau de transmission utilisé et ensuite d'émettre et de recevoir sur les lignes de transmission de ce réseau des blocs d'information conformes aux spécifications des procédures de transmission.

En plus de ces deux groupes de macro-instructions disponibles en langage d'assemblage et qui constituent la version de base du SGT, on dispose d'instructions optionnelles facilitant la gestion des zones tampon, les relevés statistiques ou les transcodages. On peut, enfin, adjoindre un module de gestion du "multitasking".

L'option multitasking autorise :

- l'utilisation de programmes réentrants pouvant traiter les échanges avec plusieurs lignes,
- le traitement parallèle de plusieurs tâches du fait de la simultanéité entre les entrées/sorties aussi bien classiques que transmission à l'intérieur d'une même partition,
- la gestion de zones de garage de paramètres ("files d'attente") associées à chaque tâche et permettant la communication entre tâches,
- le partage entre plusieurs tâches de sous-programmes non-réentrants.

Nous passerons rapidement en revue les macro-instructions du Système de Gestion de Transmission : avant toute chose, il faut demander l'incorporation des modules du SGT. C'est le rôle de la macro-instruction TRANSMOD.

Puisque les lignes ayant les mêmes caractéristiques d'équipement et d'exploitation peuvent être rassemblées en groupes, il convient de décrire chaque groupe au moyen de l'instruction LGD (Line Group Description) ; une table de même nom est alors engendrée. Une table de contrôle LCB (Line Control Block) étant associée à chaque ligne on la décrit à l'aide de la macro-instruction LCB.

Toutes les informations nécessaires au fonctionnement de la ligne y compris les paramètres fournis par l'utilisateur lorsqu'il effectue une instruction d'accès sont consignées dans cette table.

C'est elle également qui contient, en option Multitasking, les renseignements indispensables à la gestion de la tâche associée à la ligne et,entre autres, sa file d'attente.

Pour définir les adresses physiques des terminaux et des composants,on engendre une table appelée suivant les cas "liste ouverte" ou "liste fermée" contenant les caractères nécessaires pour le polling ou la sélection d'un terminal ou d'un composant de terminal.

Lorsqu'on travaille avec une liste fermée,le polling reprend en début de liste si aucun terminal n'a de message à envoyer ; dans le cas d'une liste ouverte le polling s'arrête en fin de liste.

La définition d'une liste se fait au moyen de la macro-instruction TRMLST.

Il n'y a pas de liaison biunivoque entre une liste et une ligne.

La première macro-instruction exécutable de SGT pour un groupe de lignes doit être OPEN qui permet de préparer l'emploi du groupe c'est-à-dire d'effectuer certaines initialisations des tables et l'affectation éventuelle des lignes à la partition.

Il est important de noter que la macro-instruction OPEN ne se déroule pas en parallèle avec le traitement du programme écrit dans la partition ; le contrôle n'est passé à l'instruction qui suit OPEN que lorsque les fonctions de cette dernière sont complètement traitées.

L'affectation d'une ligne à une partition lorsque ceci n'a pas été fait par OPEN,ou bien la reprise en cas d'erreur grave sur la ligne est obtenue grâce à la macro-instructions LOPEN.

Les entrées/sorties sont commandées par les macro-instructions SEND et RECEIVE.

La première permet la transmission d'un bloc vers un terminal ; deux macro-instructions SEND ou SEND et RECEIVE doivent obligatoirement être séparées par une macro-instruction de contrôle du groupe CHECK,sauf dans le cas de non lancement de la première.

La macro-instruction RECEIVE permet la réception d'un bloc depuis un terminal. Comme pour SEND l'emploi d'une macro-instruction de contrôle est obligatoire ; sans multitasking on dispose de la macro-instruction CHKM permettant de mettre le programme en attente jusqu'à ce qu'un événement parmi n se soit produit.

Ces événements peuvent être indifféremment des fins de transmission, des fins d'opération SGF (physique uniquement), des fins d'E/S demandées par TYPE ou des fins de chargement de segments.

On a la possibilité de modifier une liste de polling ou de sélection en effectuant la macro-instruction MDFLST.

D'autre part, un polling négatif utilisant une liste fermée peut être arrêté par la macro-instruction RSTPL (ReSet POLling)

Pour terminer les opérations sur les lignes de transmission d'un groupe et pour les libérer le cas échéant,on dispose de la macro-instruction CLOSE qui ne doit être programmée que lorsque les transactions avec les lignes sont effectivement terminées.

La libération d'une ligne isolée peut être obtenue par la macro-instruction LCLOSE.

Comme pour OPEN et LOPEN il n'y a pas de simultanéité possible.

Le SGT fournit également d'autres macro-instructions :

BUFPOOL : réservation de l'espace nécessaire à la création d'un pool de zones tampons et création d'un bloc de contrôle de pool,

TRMERB : création d'une table où seront comptabilisées les erreurs éventuelles de transmission,

DFSBRT : définition d'un sous-programme partageable et création du bloc-événement nécessaire à sa protection,

TRANSLATE : traduction d'un message situé dans une zone tampon de l'utilisateur ou dans une chaîne de zones tampon fournie par la gestion automatique,

GETBUF : prélever dans le pool de zones tampon affecté à un groupe de lignes un certain nombre de zones tampon reliées par des liens de chaîne,

RLSBUT : rendre une chaîne de zones tampon à son pool d'origine,

TRMSTP : imprimer sur la console de contrôle les statistiques concernant une ligne et, éventuellement, remettre à zéro les compteurs cumulatifs et les macro-instructions optionnelles du multitasking.

Comme pour le SGF il faut remarquer que le Système de Gestion de Transmission n'est pas tout à fait adapté aux exigences de notre programme.

Une première difficulté apparaît pour le chaînage des données lors de leur envoi en ligne. Le chaînage explicitement prévu dans la macro-instruction SEND concerne les zones tampon d'un pool géré automatiquement. Ceci n'aurait rien de gênant s'il n'était pas admis dans la définition du pool que toutes les zones en faisant partie sont de même longueur. Le pool est constitué d'un en-tête et de zones tampon.

adresse première zone disponible du pool	N2	N3	L = 8(N_1+ 2)

en-tête du pool

N1 : longueur de chaque zone tampon en double mots

N2 : nombre de zones tampons dans le pool

N3 : nombre courant de zones tampon disponibles dans le pool

Figure 50

Le format d'une zone tampon est représenté sur la figure ci-dessous :

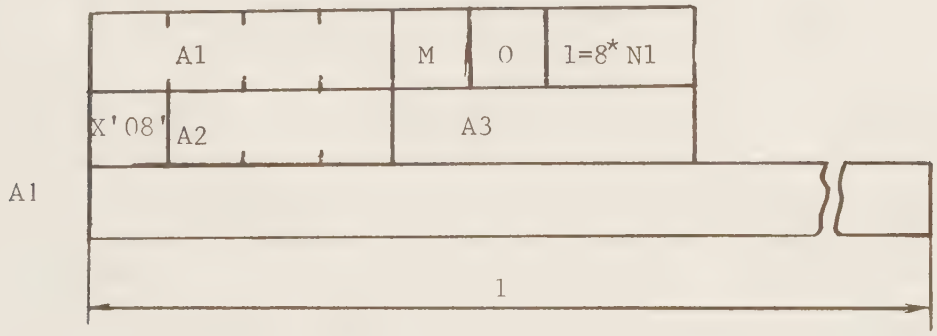


Figure 51

A1 : adresse du début de la zone utilisateur

M : masque

l : longueur utile de chaque zone tampon

X'08' : ordre de branchement d'Unité d'Echanges

A2 : adresse de l'en-tête de la zone tampon suivante dans la chaîne

A3 : adresse de l'en-tête du pool.

Si dans l'exécution de la macro-instruction SEND en option AUT (gestion automatique des zones tampon) il n'est pas fait de contrôle sur les informations stockées dans les en-têtes,on pourrait réaliser le chaînage en plaçant des données convenables dans ces derniers.

On créerait une zone des liens de chaîne comportant deux doubles mots par zone tampon chaînée :

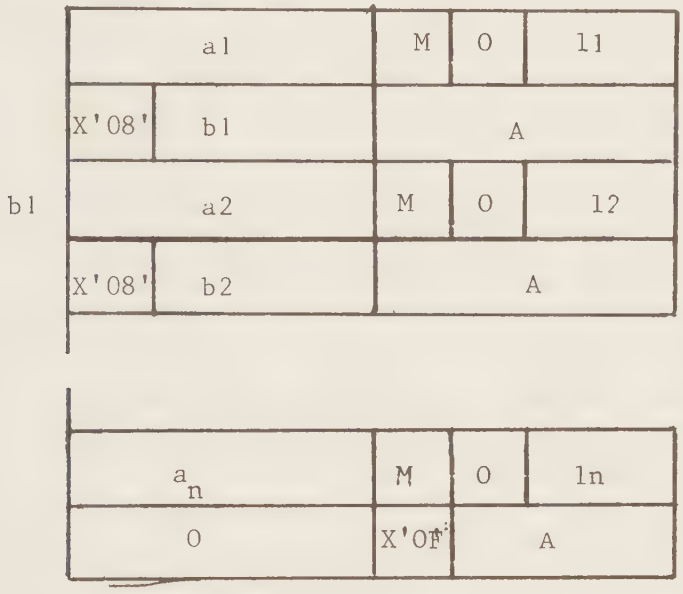


Figure 52

La dernière entrée doit comporter un ordre d'arrêt d'Unité d'Echange.

Une deuxième difficulté surgit lorsqu'on veut utitiliser la décomposition du programme en tâches et leur gestion par le Multitasking du SGT.

En effet, une tâche dans le SGT est obligatoirement liée à une ligne et à une ligne on ne peut faire correspondre qu'une seule tâche.

Les tâches sont créées avec une priorité qui est fixe et ne peut pas être changée au cours de la vie de la tâche.

Tout ceci nous a amené à renoncer à l'emploi du Multitasking SGT et à concevoir une autre version, plus "libérale", nous semble-t-il, du régulateur des tâches.

6.3 Régulateur

6.31 Notion de tâche

Nous appelons tâche une partie du programme ("bout de code") qui remplit une fonction donnée et à qui on veut accorder une certaine autonomie. Des données peuvent, évidemment, être associées à une procédure.

Une tâche est définie par l'adresse de la première instruction à exécuter.

La communication entre tâches peut se faire,entre autres, par la file d'attente qu'il est possible d'associer à la tâche.

On a la possibilité de synchroniser une tâche avec l'arrivée d'un événement parmi plusieurs attendus.

La décomposition du programme en tâches se justifie et présente un intérêt lorsque ce premier comporte des fonctions que l'on pourrait imaginer se déroulant, en partie du moins, en parallèle. Elle n'a pas beaucoup de sens dans un programme entièrement séquentiel et ne comportant aucune attente d'événements extérieurs (ou intérieurs).

La gestion des tâches est assurée par un ensemble de sous-programmes que nous désignerons sous le nom de régulateur en nous inspirant de la terminologie employée dans les spécifications du Système de Gestion de Transmission.

6.32 Fonctions du régulateur

6.321 Activation d'une tâche

Pour que l'exécution d'une tâche puisse être lancée,il faut que celle-ci ait été activée. L'activation s'obtient par un appel au sous-programme ACTIV qui ne lance pas la tâche mais seulement crée une entrée dans les tables du régulateur. Lors de l'activation on assigne une priorité à la tâche, cette priorité P est soumise à la restriction $0 \leqslant P \leqslant 255$, 0 étant la priorité minimum et 255 la priorité la plus élevée.

Plusieurs tâches peuvent avoir le même P, elles sont alors rangées dans l'ordre d'activation,la première activée étant la plus prioritaire.

Les priorités ne sont pas obligatoirement statiques mais peuvent évoluer au cours de la vie de la tâche.

Par ailleurs, on doit préciser l'adresse de la file d'attente de la tâche,ou bien indiquer que celle-ci n'en aura pas.

A l'appel au sous-programme ACTIV les registres 3 et 4 doivent contenir les paramètres de définition de la tâche :

R3 | P | nom de la tâche

nom tâche : adresse première instruction à exécuter

adresse file d'attente

R4 si une file d'attente doit être associée à la tâche

X'80' si la tâche n'a pas de file d'attente

Le registre 8 a été choisi comme registre d'appel aux sous-programmes du régulateur.

L'appel à ACTIV peut se trouver n'importe où dans le programme et en particulier dans le corps d'une tâche (activation d'une tâche par une autre).

Remarquons qu'il n'est pas possible de créer deux tâches portant le même nom.

6.322 Lancement du Multitasking

La tâche la plus prioritaire est la première à prendre le contrôle après l'exécution d'un appel au sous-programme BEGP.

Elle sera lancée à l'instruction se trouvant à l'adresse indiquée par le nom de la tâche.

On garde l'adresse de l'instruction qui suit BAL,8 BEGP ; on se branchera à cette adresse lorsqu'il n'y aura plus aucune tâche activée.

Si aucune tâche n'a été introduite avant l'appel à BEGP (cas d'erreur de programmation),on continue en séquence.

6.323 Introduction d'un mot dans la file d'attente d'une tâche

Comme nous l'avons indiqué précédemment, on a la possibilité d'associer une file d'attente à une tâche lors de son activation. D'autres tâches peuvent placer dans cette file des informations dont la tâche a besoin pour son exécution.

Les files d'attente sont gérées en FIFO (First In First Out).

Pour introduire un mot dans la file d'attente d'une tâche, on fait appel au sous-programme ENQ, en précisant dans le registre 3 le nom de la tâche au profit de laquelle doit s'effectuer l'opération et en préparant dans le registre 4 l'information que l'on désire passer. Seul le registre général 3 est modifié par le sous-programme ; il contient, en effet, un code de retour permettant de savoir si le mot a pu être placé dans la file d'attente de la tâche désignée.

6.324 Prélèvement d'un mot dans la file d'attente de la tâche

Comme la file d'attente est gérée en FIFO, une tâche ne peut demander qu'à prélever le mot le plus ancien dans la file.

Ceci est obtenu par l'appel du sous-programme DEQ du régulateur (registre d'appel R8). Le registre 3 contient le code de retour. S'il n'y a pas eu d'erreur de programmation et si la file n'est pas vide, le mot prélevé se trouve dans le registre 4. Dans tous les cas le contrôle est rendu à l'instruction qui suit BAL,8 DEQ.

6.325 Attente d'un événement parmi n

Pour pouvoir mettre une tâche en attente d'un événement parmi n, il faut avoir défini une table des événements attendus (TEA). Elle doit comporter un double mot par événement susceptible de se produire.

On distingue trois types d'événements :

- fin entrée/sortie classique (événement SGF)
- fin entrée/sortie transmission (événement SGT)
- fin entrée/sortie demandée par la macro-instruction TYPE du moniteur, fin de chargement de segment ou tout autre événement remettant à 0 l'élément binaire 0 d'un bloc événement (événement TYPE).

Il est possible d'ignorer une entrée en lui assignant un type spécial.

A chaque événement on associe un bloc événement dont le premier

octet indique si l'événement est arrivé ou non :

X'80' : événement non arrivé

X'00' : événement arrivé.

La mise en attente éventuelle est obtenue par un appel au sous-programme ATMLT du régulateur (BAL,8 ATMLT). Le registre général trois doit contenir les paramètres d'appel :

n : nombre d'événements susceptibles de se produire.

Il faut souligner qu'une tâche qui effectue un appel à ATMLT peut perdre le contrôle même si un événement parmi ceux qu'elle attendait est arrivé.

En effet, on commence par préserver les douze premiers registres de la tâche ; on parcourt ensuite les tables du régulateur en cherchant la tâche la plus prioritaire prête à être relancée. Si aucune tâche n'est prête,on se bloque en attente au moyen de la macro-instruction MLTW (Wait for Multiple events) du moniteur ; la tâche à laquelle appartient le premier événement arrivé reprendra le contrôle.

n peut être égal à 0 ; on considère alors que la tâche n'attend aucun événement ; elle va continuer à l'instruction qui suit l'appel à ATMLT s'il n'y a pas de tâche plus prioritaire prête. Pour des raisons de compatibilité avec les instructions du SGT,nous avons choisi le registre douze pour les informations de retour.

Si tout s'est bien passé,R12 contiendra l'adresse absolue du mot de la table des événements attendus pointant vers le bloc événement dont le chiffre binaire 0 est à 0.

Il est important de bien déterminer les événements attendus et les priorités des tâches,si l'on veut éviter que les tâches se bloquent toutes en attente d'événements internes ("étreinte fatale") ou bien encore qu'une tâche monopolise l'unité centrale.

La tâche relancée continue en séquence, ses registres R0 et R11 reprennent la valeur qu'ils avaient au moment de l'appel à ATMLT.

La file d'attente qu'il est possible d'associer à une tâche est conçue de façon que le premier mot de la file constitue un bloc événement lié à la condition 'file d'attente vide'.

Une tâche peut donc facilement attendre que des informations aient été placées dans sa file d'attente.

Soulignons, enfin, que la notion de tâche n'a pas grand chose à voir avec celle de processus et de processeur virtuel.

Le régulateur n'interrompt une tâche que si BAL,8 ATMLT a été programmé. Une erreur de conception commise dans un programme peut bien conduire à ce qu'une tâche, quoique activée, ne prenne jamais le contrôle et ne s'exécute donc jamais.

6.326 Changement de la priorité d'une tâche

Il est souvent utile et parfois indispensable d'effectuer une action avant une autre. Si ces actions constituent des tâches séparées, ceci peut être obtenu en assignant des priorités convenables aux tâches au moment approprié.

Il suffit pour cela de charger le registre général trois du nom de la tâche et de sa nouvelle priorité et de faire appel au sous-programme ASPRIOR du régulateur.

Le même registre trois contiendra un code de retour permettant de savoir si tout s'est bien passé.

Le changement de priorité consiste à mettre à jour les tables du régulateur ; il peut être effectué par une tâche pour son propre compte ou bien au profit d'une autre tâche.

Dans tous les cas, le contrôle est rendu à l'instruction qui suit BAL,8 ASPRIOR.

La tâche est, en quelque sorte, réactivée avec une nouvelle priorité (on applique les mêmes règles que pour ACTIV) ; il n'est pas possible de créer la tâche dans ASPRIOR, le changement de priorité concerne obligatoirement une tâche qui a déjà fait l'objet d'une activation (BAL,8 ACTIV) auparavant.

6.327 Désactivation d'une tâche

Lorsqu'une tâche a terminé son travail,il faut la faire disparaître des tables du régulateur ou autrement dit la désactiver, ce qui est obtenu par un appel au sous-programme DSACTV.

Une tâche peut demander indifféremment sa propre désactivation ou celle d'une autre tâche.

Si la désactivation concerne une autre tâche que celle lancée, on efface simplement les entrées correspondantes dans les tables et le contrôle est rendu à l'instruction qui suit BAL,8 DSACTV.

Si, par contre, la tâche se désactive elle-même, on met à jour les tables du régulateur et on relance la tâche la plus prioritaire prête. Lorsqu'il ne reste plus de tâches activées, on considère que le régulateur a terminé son travail et on se branche à l'instruction qui suit l'appel au sous-programme de lancement du régulateur (BEGP).

6.328 Remarques sur le régulateur

Ainsi, un programme écrit en termes de tâches aura l'allure suivante :

```
          CSECT
     *ACTIVATION DES TACHES
          LD4I,3     T1          NOM TACHE
          LD1I,3     P           PRIORITE
          LD4I,4     FA1         ADRESSE FILE D'ATTENTE
          BAL,8      ACTIV
          LD4I,3     T2
            :
            :

     *DEBUT DU MULTITASKING
          BAL,8      BEGP
     *FIN DU MULTITASKING
          TERM
     *DEFINITION DES TACHES
T1        BOUND       4
            :
            :
          LD4I,3     TEA 1       ADRESSE TABLE EVENEMENTS ATTENDUS
          LD1I,3     N1          NOMBRE EVNTS POSSIBLES
          BAL,8      ATMLT       ATTENTE
            :
            :
```

```
            LD4I,3      T1          NOM TACHE
            BAL,8       DSACTV      LA TACHE SE DESACTIVE
T2          BOUND       4
            LD4I,3      TEA2
            LD1I,3      N2
            BAL,8       ATMLT
              :
              :
            LD4I,3      T2          NOM TACHE A DESACTIVER
            BAL,8       DSACTV
              :
              :
```

Il est possible de rendre le code d'une procédure (tâche) partageable.

Il faut pour cela qu'elle soit réentrante, ce qui peut être obtenu en utilisant les sections fictives et en gérant convenablement les registres de base. La notion de file d'attente facilite les choses parce qu'on ne spécifie pas l'adresse de la file en y prélevant 1 mot.

Il reste la difficulté liée au fait qu'il est interdit d'activer deux tâches portant le même nom.

Elle peut être facilement levée, si l'on donne comme l'adresse de la première instruction (nom de la tâche) l'adresse d'une instruction de branchement au début de la procédure proprement dite.

Remarquons, enfin, que les macro-instructions du SGF (et du SGT) n'étant pas réentrantes, il faut absolument éviter que deux tâches accèdent simultanément au même fichier. L'une des façons de le faire consiste à regrouper les instructions de traitement du fichier et à leur adjoindre un bloc événement, de manière à savoir à tout moment si cette séquence d'instructions est disponible ou si, au contraire, elle est en cours d'exécution au profit d'une autre tâche, auquel cas il faut attendre qu'elle redevienne libre avant de pouvoir l'exécuter.

Les détails d'implémentation et le mode d'emploi précis du régulateur ont été donnés en annexe.

6.4 Description du programme réalisé

6.41 Généralités

Nous avons pensé qu'il était intéressant de profiter au maximum du travail d'étude fait dans les parties précédentes de l'exposé.

Il n'est malheureusement pas possible de transposer tout sans modification, essentiellement parce que nous ne disposons pas d'un système à processeurs virtuels.

Nous avons gardé la décomposition du programme en trois parties :

- communication avec l'utilisateur (l'opérateur)
- émission
- réception.

Dans la mesure du possible nous avons voulu faire correspondre à chaque processus une tâche.

Le régulateur nous fournit un moyen de synchronisation : blocs événements et attente d'un événement parmi n.

Nous nous sommes attachés à une définition précise et rationnelle des interfaces software entre les tâches et à une paramétrisation du programme,de façon à garder en paramètre la plupart des valeurs susceptibles de changer ce qui doit permettre, le cas échéant, de modifier uniquement une déclaration en début de section sans avoir à retoucher au corps du programme.

D'autre part, dans le système IRIS 50 disponible actuellement à l'IRIA la partie "transmissions" n'est pas implémentée. Nous avons par conséquent décidé de simuler,en premier temps, l'émission et la réception en ligne en "court-circuitant", en quelque sorte, cette dernière de façon que le programme est mis en correspondance avec lui-même. Il fonctionne alors en programme de conversion de support (cartes - imprimante).

Comme nous l'avons dit, le programme a été décomposé en un certain nombre de tâches ; nous les décrirons maintenant en détail.

6.42 Entrée des messages sur la machine à écrire de pupitre

6.421 Interruption pupitre

Nous avons choisi la machine à écrire de pupitre pour la communication avec l'utilisateur. Pour donner à l'utilisateur le possibilité de rentrer son message à tout instant,nous avons décidé d'utiliser l'interruption pupitre.

En fonctionnement habituel du système SIRIS 2,cette interruption constitue une condition d'exception.

On peut cependant demander à traiter les conditions d'exception en établissant "un retour exception".

Nous disposons à cette fin de la macro-instruction SXR (Set eXception Return) qui, lorsque l'événement y cité se produit,permet de se brancher à une séquence de retour exception. Cette séquence (voyez fig. 53) contient essentiellement une remise à zéro du bloc événement BERM (Bloc Evénément-Réception du Message mae),ce qui signalera à la tâche DMMAE qu'une intervention opérateur a eu lieu ; on retourne ensuite dans la tâche interrompue en validant le retour exception au moyen de la macro-instruction VXR (Validate eXception Return).

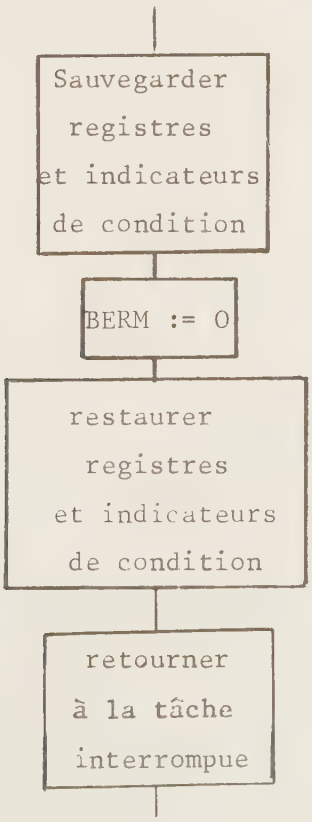


Figure 53.

Séquence de retour exception sur "intervention opérateur".

6.422 Syntaxe du langage de commande

Nous distinguons deux sortes de messages qui peuvent être tapés à la MAE :

- messages destinés à être transmis à l'opérateur de l'ordinateur-correspondant,
- messages "locaux" s'adressant au programme (commandes).

Seul le contenu de ces dernières est susceptible d'avoir une signification pour notre programme, c'est pourquoi tout message destiné à la transmission doit obligatoirement commencer par le caractère ':' les messages locaux étant repérés par '*'.

Nous avons adopté pour ces commandes la syntaxe suivante :

<commande> ::= <nom de commande> [<liste de paramètres>]
<liste de paramètres> ::= <paramètre> [<liste de paramètres>]
<paramètre> ::= <blanc> <mot clé> [<blanc>] <assignateur> [<blanc>]<argument>
<assignateur> ::= :
<argument> ::= <nombre> | <identifieur>

C'est un système à base de mots clés permettant de mettre les paramètres dans un ordre quelconque.

Voici un exemple des deux classes de messages :

: MONTEZ LA BANDE 1020 message de transmission

↑
1
↓

* FERMER commande

La liste des commandes avec les paramètres propres à chacune d'entre elles a été donnée en annexe.

6.43 Interprétation du langage de commande

L'interprétation des messages de l'opérateur a été confiée à la tâche DMMAE (Décodage Messages Machine A Ecrire) qui doit être de priorité élevée si l'on veut obtenir une réaction rapide du programme. DMMAE commence par tester le premier caractère du message reçu ; si c'est ':',on met à "événement arrivé" (RAZ) le bloc événement BEMAE (Bloc EvénementMachine A Ecrire) qui signale à la tâche de formation du message ligne (FML) qu'un élément pour le sous-canal MAE doit être

pris en compte. Si le premier caractère est un astérisque,on essaie d'interpréter la commande.

La tâche dispose à cet effet d'un tableau où l'on a rangé les noms des commandes admises. Lorsque l'une d'entre elles a été reconnue, on se branche à la séquence correspondante.

Dans le cas contraire, ou bien si le premier caractère n'est pas : ou *, un message d'erreur est placé dans la file d'attente de la tâche de sortie des messages à la machine à écrire ECRMAE. Cette façon de procéder permet d'étendre à volonté le langage de commande ou de modifier les paramètres d'une commande,sans avoir à réorganiser l'ensemble de la tâche DMMAE. Il suffit d'ajouter une entrée dans le tableau des noms de commande et une séquence d'interprétation de la commande.

Au cas où la syntaxe de la commande n'est pas conforme à celle spécifiée, un message d 'erreur sera introduit dans la file d'attente de ECRMAE.

S'il veut recommencer, l'utilisateur devra appuyer de nouveau sur le bouton interruption. Nous avons représenté l'algorithme de la tâche sur la figure 54 ci-après.

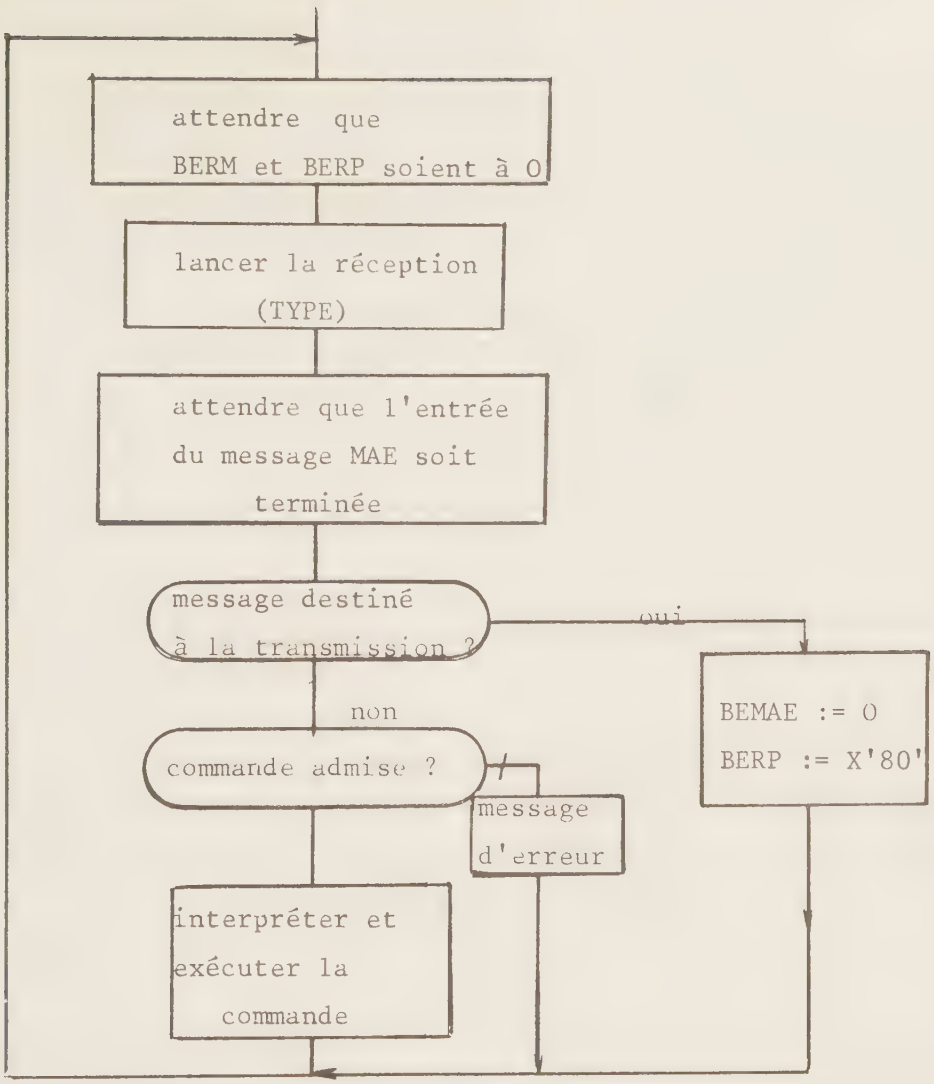


Figure 54. Tâche DMMAE - Interprétation des messages de la machine à écrire.

DMMAE communique avec les autres tâches par l'intermédiaire des blocs événements suivants :

BERP - le tampon de réception des messages MAE est libre

BERM - il y a eu une intervention opérateur

BEMAE - message à transmettre attend dans le tampon TMAER

BECAV - bloc événement indiquant si des commandes ont été placées dans la zone des commandes à envoyer.

La macro-instruction TYPE se réfère au bloc de commande TWBENT ; elle permet l'entrée d'un message à la machine à écrire.

6.44 Lecture dans le fichier cartes

La tâche LECRT de lecture dans le fichier (voir fig. 58) a pour rôle de former un élément dans le tampon composant. Au moment du "réveil"

la tâche trouve dans sa file d'attente les renseignements qui lui sont indispensables :

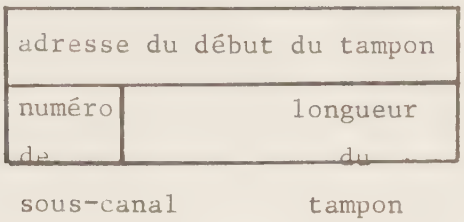


Figure 55. Image de la file d'attente (FACRT) de LECRT au réveil

Une fois le tampon composant rempli,le sous-canal est placé dans la file des sous-canaux prêts - FILEP ; on remet en même temps à 0 le bloc événement BENSC (Bloc Evénement Nouveau Sous-Canal).

Voici l'organisation de la file :

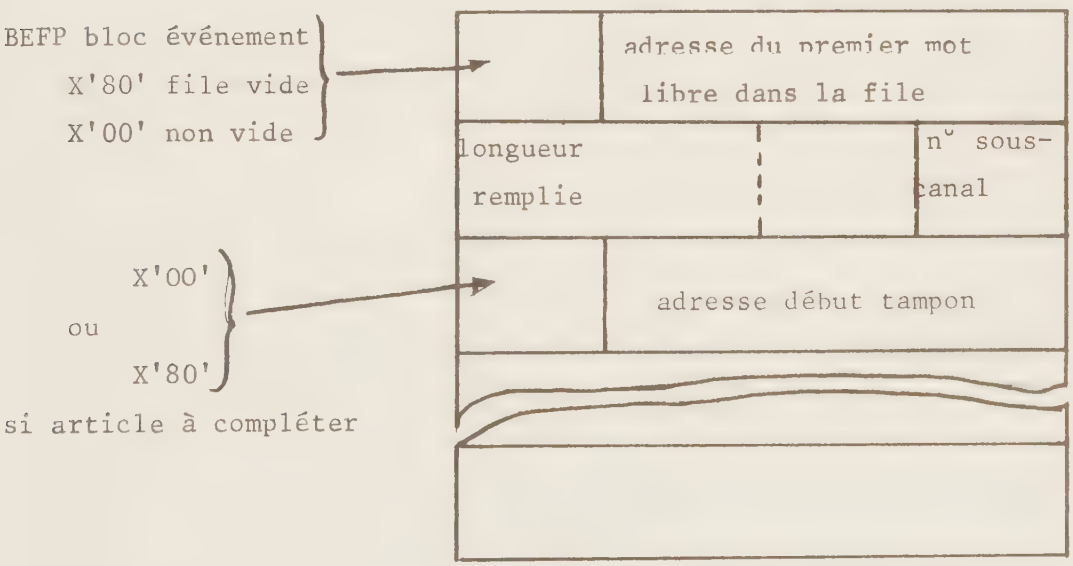


Figure 56. File des sous-canaux prêts FILEP

Chaque entrée dans la file comporte deux mots ; le premier donne la longueur en nombre d'octets de l'élément et le numéro de sous-canal ; le deuxième précise l'adresse du début de l'élément.

L'indicateur "article à compléter" (X'80') n'est placé que par la tâche de formation du message ligne FML lorsqu'un article à dû être découpé pour les besoins de la transmission.

Au moment où un sous-canal entre dans la file cet octet est toujours à 0.

Le bloc événement BEFP (Bloc Evénement de la file P) est associé à la condition "FILEP vide". Si une fin de fichier a été détectée, on en garde la trace dans la table FFCH en y plaçant le numéro du sous-canal correspondant :

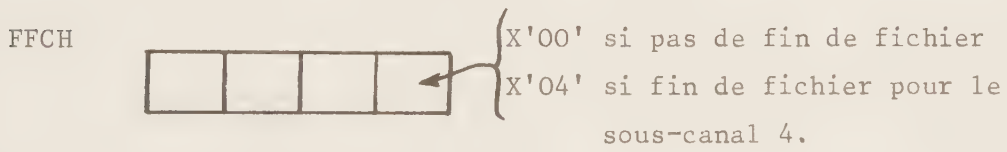
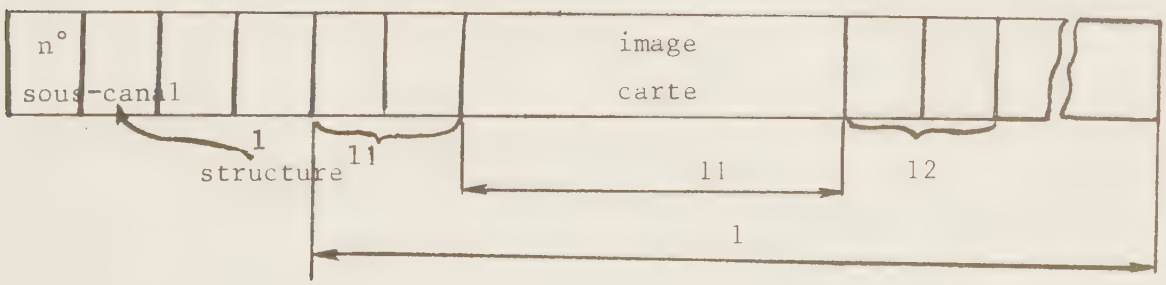


Figure 57. Table des fins de fichier FFCH

L'image du tampon composant à la sortie de la tâche est montrée ci-dessous :



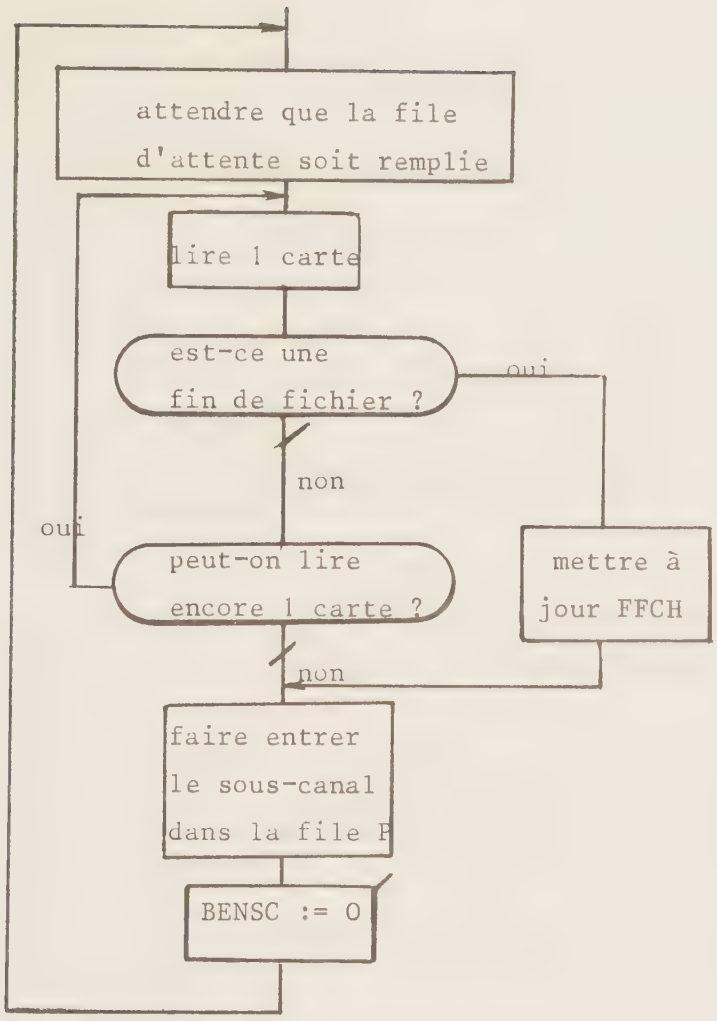


Figure 58. Tâche LECRT - algorithme

En résumé,la tâche LECRT communique avec les autres tâches par l'intermédiaire des blocs événements suivants :

- sa file d'attente FACRT
- BENSC : nouveau sous-canal entre dans la file P
-BEFP : file des sous-canaux prêts FILEP non vide.

6.45 Formation du message de transmission en ligne

Dans la tâche FML (Formation du Message Ligne) on prépare essentiellement les données pour le chaînage dans les commandes canal (algorithme - figure 62).

Ces données sont consignées dans la table TCHAINE dont voici la composition :

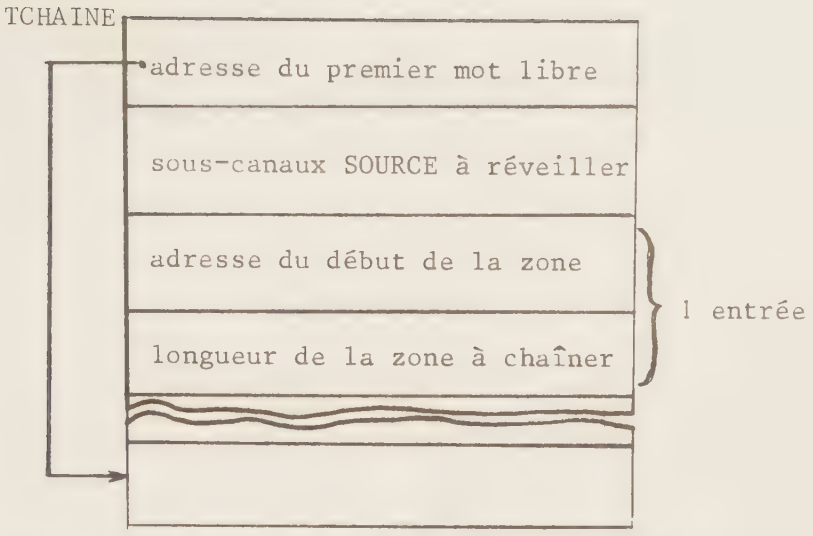


Figure 59. Table des liens de chaîne TCHAINE

Chaque entrée dans TCHAINE comporte deux mots dont le premier contient l'adresse du début de la zone à chaîner et le second sa longueur.

La tâche commence par attendre que l'un des événements suivants soit arrivé :

- il y a des commandes à envoyer dans la zone des commandes à envoyer COMAENV
- il y a un élément pour le sous-canal MAE
- un accusé de réception positif ACC a été reçu ou généré depuis la dernière exécution de la tâche
- un nouveau sous-canal est entré dans la file des sous-canaux SOURCE prêts
- la file FILEP n'est pas vide.

Ce dernier événement est ignoré si :

- le sous-canal élu se trouvait en queue dans FILEP,
- il n'y a pas eu de ACC'S pour les sous-canaux prêts ou pas de ACC'S du tout.

Si on ne le faisait pas, la tâche monopoliserait l'unité centrale, à moins d'être de priorité inférieure à toutes les autres tâches du programme.

On continue en testant la présence des commandes à envoyer et de l'élément du sous-canal MAE ; des données de chaînage sont éventuellement inscrites dans TCHAINE.

On examine ensuite la file P et la table des ACC'S reçus ou générés ACCREC (fig. 60) en essayant de choisir 1 sous-canal.

Si l'élément de ce sous-canal dépasse la longueur transmissible, on le fractionne et c'est alors que l'indicateur "article à compléter" est mis à X'80' dans FILEP.

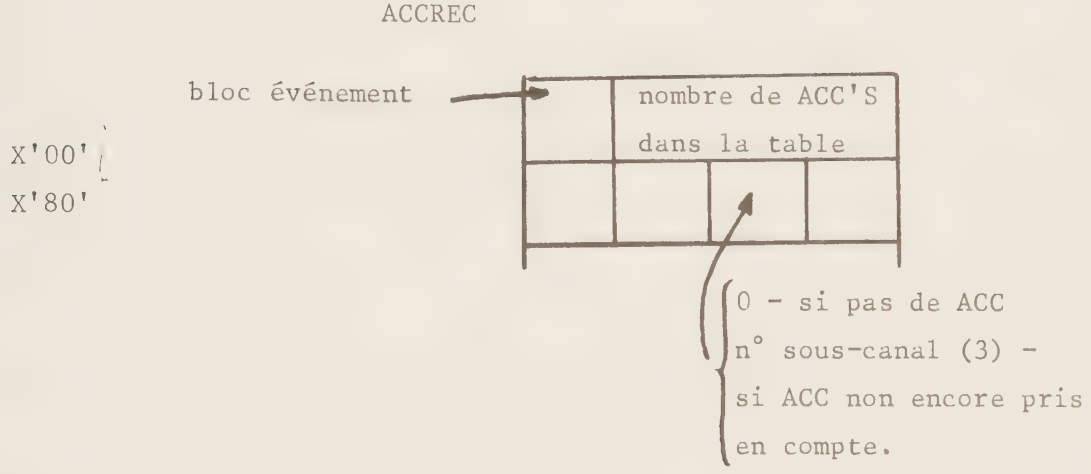


Figure 60. Table des ACC'S reçus ou générés ACCREC

Le bloc événement de la table ACCREC est remis à X'80'.

On admet que l'élément peut être soit un début, une suite ou une fin d'article, soit un article complet, soit plusieurs articles.

Lorsqu'un sous-canal a quitté la file P, son numéro est inscrit dans la partie "sous-canaux à réveiller" de TCHAINE, ce qui veut dire que la tâche de lecture du fichier sera relancée dès que le tampon composant redeviendra disponible.

Dans la version actuelle de la tâche FML on ne choisit qu'un seul sous-canal (autre que celui de communication entre les machines à écrire de pupitre) ; on multiplexe donc dans le temps. Remarquons qu'aussi bien la zone des commandes à envoyer que les tampons composant et de réception MAE se trouvent indisponibles jusqu'à la fin de l'envoi en ligne.

Si l'on veut donc accélérer le fonctionnement, il faudra prévoir deux zones qui serviront en bascule pour former les commandes à envoyer ;

lorsqu'un seul sous-canal source est actif,il sera bon également de lui associer deux tampons composant.

La tâche d'émission en ligne est relancée grâce à la remise à 0 du bloc événement EBEL.

En résumé, voici les blocs événements qui servent à synchroniser FML avec les autres tâches :

- ACCREC : un ACC reçu ou généré depuis le dernier examen de ACCREC,
- BENSC : nouveau sous-canal dans la file P,
- BEFP : file P non vide,
- COMAENV : il y a des commandes à envoyer,
- BEMAE : il y a un élément pour le sous-canal MAE,
- EBEL : le message est formé.

6.46 Emission en ligne

Ne disposant pas pour l'instant du Système de Gestion de Transmission,nous avons simulé l'émission en ligne.

La tâche est réveillée lorsque EBEL est à 0. On examine la table des liens de chaîne TCHAINE. Les parties composant le message sont transférées dans le tampon de réception ligne TARL de la partie réception. La réception du message sur la ligne est donc simulée en même temps.

Une fois TARL rempli on réveille le sous-canal et l'on autorise une nouvelle entrée de message à la machine à écrire en remettant à 0 le bloc événement BERP (Bloc Evénement Réception Permise).

La table TCHAINE est remise à jour.

Les renseignements concernant les sous-canaux SOURCE DE DONNEES sont consignés dans la table TSCAS (Table des Sous-Canaux Actifs Source) dont la composition est représentée sur la figure ci-après.

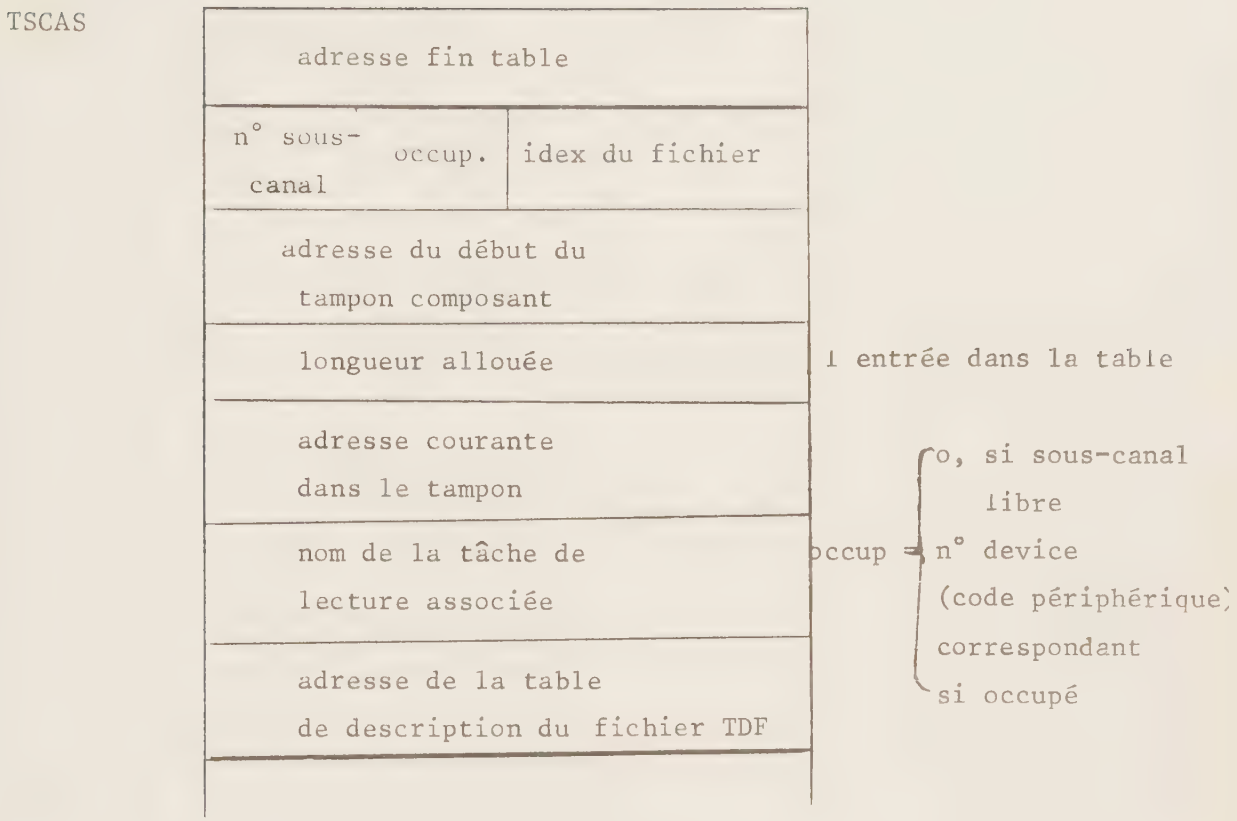


Figure 61. Table des sous-canaux actifs source

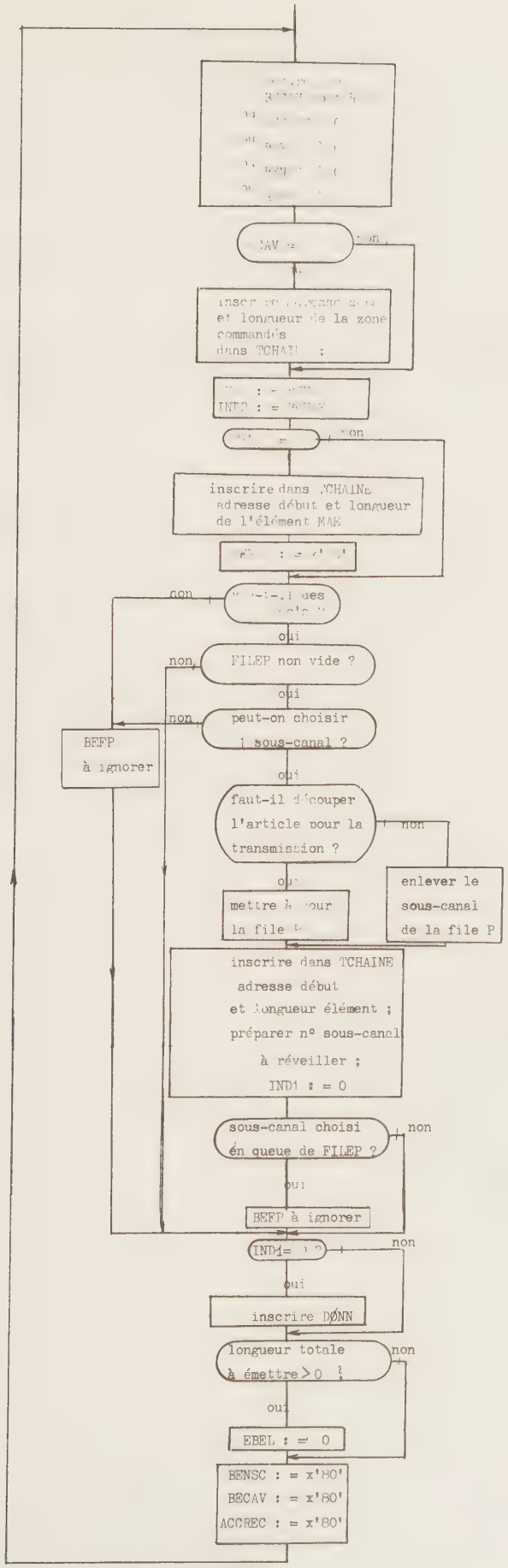


figure 62. Tâche FML de formation du message d'émission en ligne

TSCAS comporte 1 entrée par sous-canal source actif ; elle est initialisée lors de l'établissement du sous-canal.

L'idex du fichier est l'identifieur d'exploitation par lequel on repère le fichier dont la description se trouve dans la TDF associée.

On indique également le nom de la tâche de garnissage du tampon composant (lecture dans le fichier).

6.47 Réception et interprétation du message de transmission

Rappelons tout d'abord la composition du message reçu :

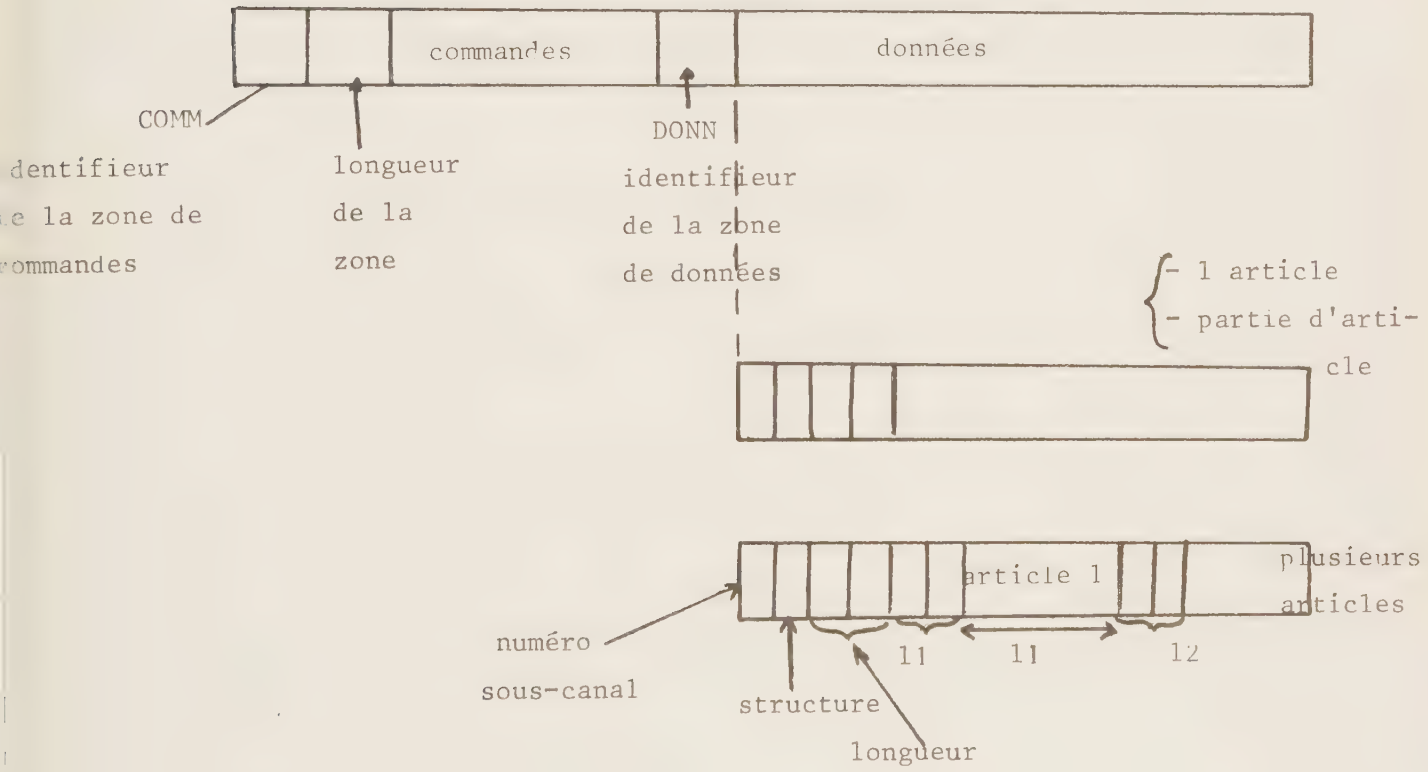


Figure 63

Nous admettons que la zone de commandes se trouve toujours (si elle est présente) en tête du message. Lorsqu'un message a été reçu,la tâche TRL (Tache de Réception sur la Ligne) commence par examiner le premier octet du message ; si celui-ci est COMM,on interprète le contenu de la zone de commandes. La rencontre de DONN signale qu'il s'agit de la partie "données" du message. Suivant la valeur de l'octet "structure" on transfère dans le tampon composant du sous-canal PUITS la partie utile de l'élément.

Si le tampon composant contient maintenant 1 article complet ou plusieurs articles,on cherche dans la table des sous-canaux actifs PUITS le nom de la tâche de vidage du tampon correspondant (écriture dans le fichier).Chacune de ces tâches possède une file d'attente ; on y place l'adresse du début de l'élément comptée à partir de l'octet structure et le numéro du sous-canal :

Figure 64. Image de la file d'attente d'une tâche de sortie au moment du réveil.

TSCAP est analogue à la table des sous-canaux Source Actifs comme on peut le constater sur la figure ci-après:

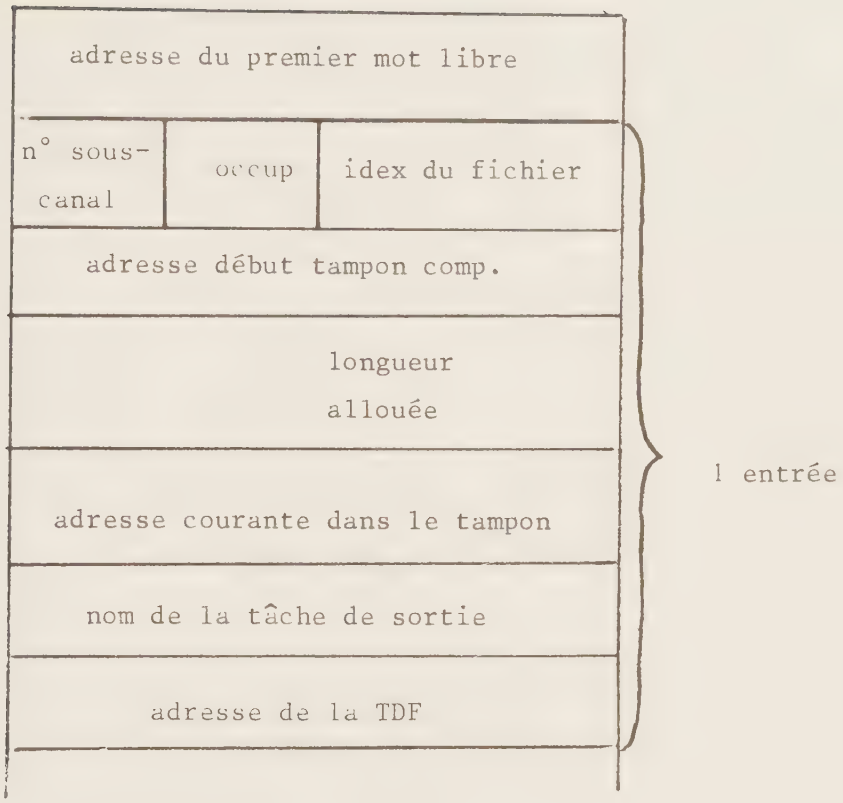


Figure 65. Table des Sous-Canaux Actifs Puits

On retrouve ainsi l'octet "occup" qui indique quel composant est utilisé en sortie, l'identifieur d'exploitation du fichier (idex), les renseignements concernant le tampon composant, le nom de la tâche qui se charge de son vidage et, enfin, l'adresse de la table de description du fichier TDF.

L'élément du sous-canal MAE est traité un peu à part ; on le reconnaît à son numéro du sous-canal qui lui est réservé ; il est transféré dans un tampon spécial TMAES ; l'adresse de ce tampon est ensuite placée dans la file d'attente de la tâche ECRMAE de sortie des messages sur la MAE.

Le fonctionnement avec ligne réelle pourrait être celui de la figure 66.

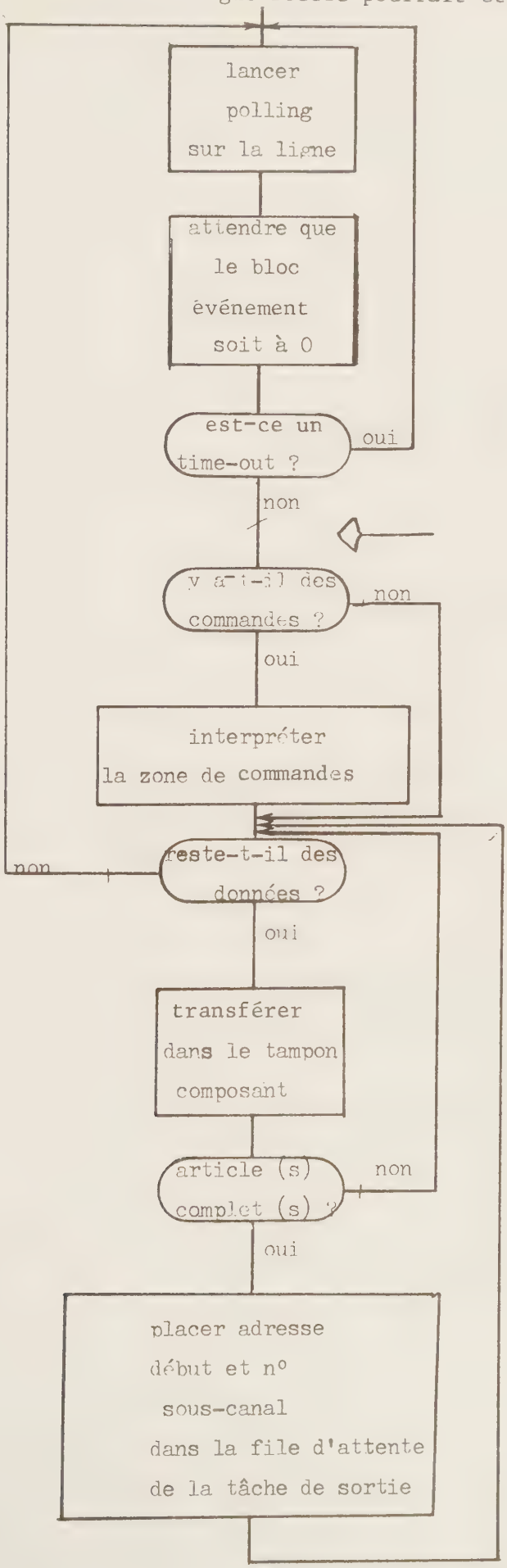


Figure 66

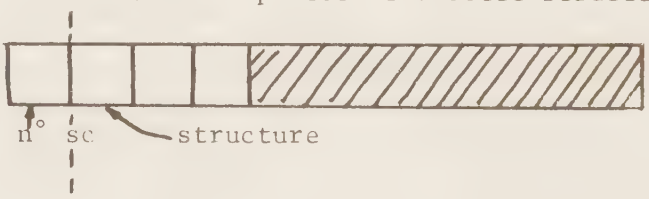
Dans la version "court-circuitant" la ligne on se branche à l'endroit marqué par la flèche après une attente que le bloc événement BEDBL (Bloc Evénement Décodage Bloc Ligne) soit à 0.

La tâche se sert donc de BEDBL et des files d'attente des tâches de sortie comme moyen de synchronisation mutuelle.

La façon dont on transfère l'élément dans le tampon composant varie suivant la structure de ce dernier :

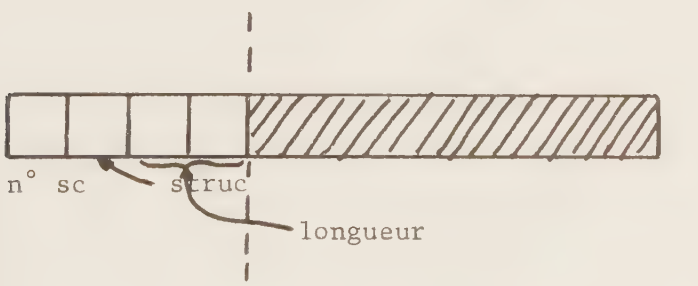
- début d'article
- plusieurs articles
- 1 article complet

on transfère à partir de l'octet structure



- suite d'article
- fin d'article

on transfère à partir de la partie d'article



structure tampon correspondant

on remet à jour la longueur totale dans le tampon composant.

Figure 67. Reconstitution des articles dans les tampons composants à la réception.

6.48 Ecriture dans le fichier imprimante.

Nous parlerons maintenant de la tâche de sortie sur l'imprimante ECRIMPR puisque nous avons choisi l'imprimante comme composant PUITS dans notre exemple.

Au réveil, la tâche trouve dans sa file d'attente l'adresse de l'octet structure dans le tampon composant ainsi que le numéro du sous-canal auquel l'élément appartient.

On ne réveille une tâche de sortie que si l'article a été reconstitué et donc son octet structure ne peut valoir que "1 article" ou "plusieurs articles".

L'utilisation raisonnable correspond à des articles de longueur inférieure à la capacité de l'imprimante qui est de 132 caractères ; on a néanmoins prévu le cas où la longueur d'un article dépasserait cette limite.

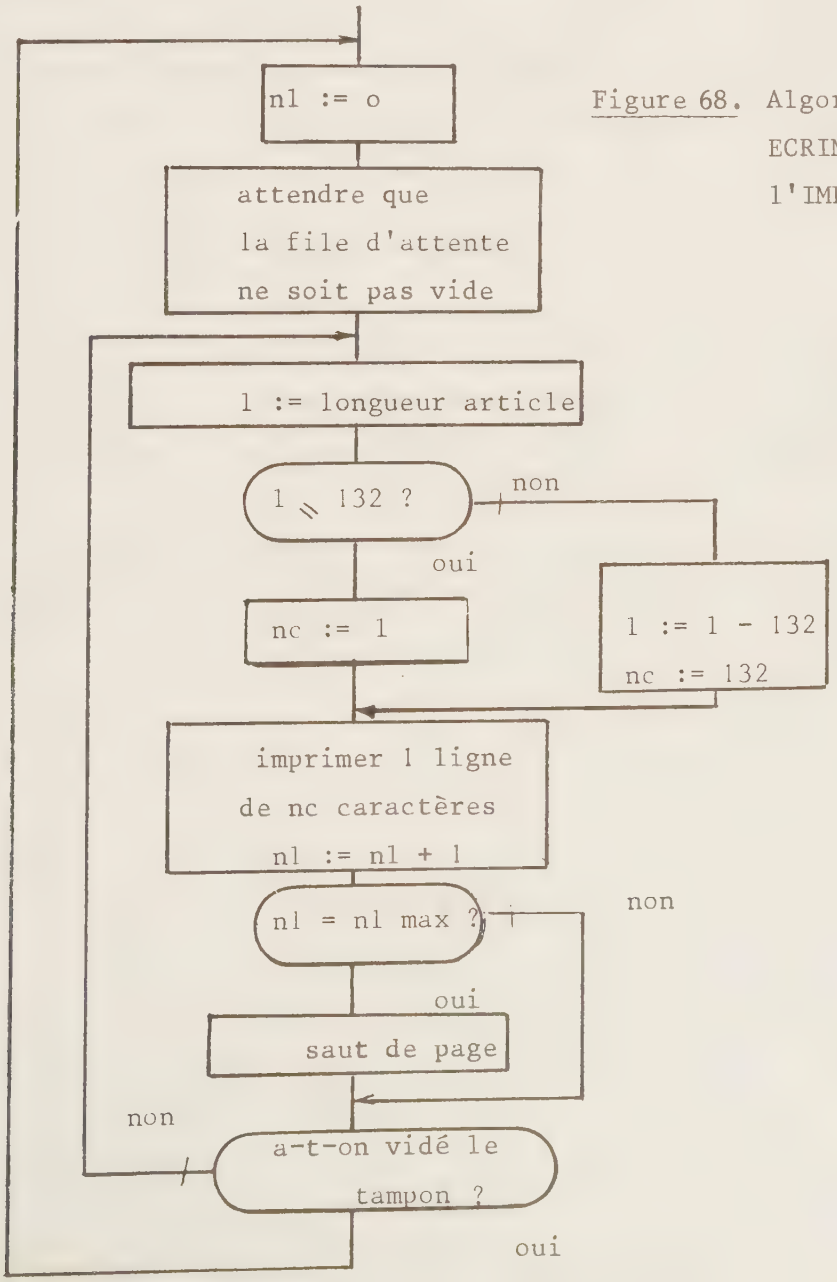
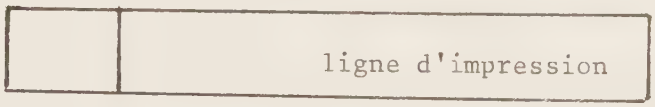


Figure 68. Algorithme de la tâche ECRIMPR (ECRiture sur l'IMPRimante).

Les interlignes et les sauts de page sont commandés par un octet de saut S placé devant le premier caractère à imprimer.



octet
de saut

Si tout s'est bien passé,la tâche prépare un accusé de réception positif ACC pour le sous-canal.
Ce ACC est introduit dans la zone des commandes à envoyer COMAENV.
Dans un fonctionnement non simulé il faut soit utiliser deux zones en bascule soit interdire l'accès à COMAENV jusqu'à la fin de l'émission en ligne,en associant à la zone un bloc événement de protection.

6.49 Remarques finales.

L'exemple d'application de la procédure et les premiers résultats obtenus semblent montrer que l'ensemble des commandes et la structure des informations définis permettent effectivement l'établissement et le fonctionnement d'un sous-canal.
Seule une utilisation véritable avec des lignes de transmission peut montrer les inconvénients et les avantages de notre procédure d'exploitation.

ANNEXE 1

Justification de l'algorithme d'attente d'un événement parmi n. Nous supposons que chaque événement ne peut se produire à nouveau tant qu'on n'a pas effectué le traitement associé.

La situation peut se résumer comme suit :

```
"begin
 integer S, P1, P2, flag 1, flag 2 ;
  flag 1 := 0 ;
 flag 2 := 0 ;
  S := 0 ;
  P1 := 0 ;
  P2 := 0 ;
         parbegin
  T   :      begin
             P(S) ;
             V(S) ;
             if flag 1 = 1 then
                  begin
                   flag 1 := 0 ;
                  P(S) ;
                  V(P1) ;
                  end ;
             if flag 2 = 1 then
                  begin
                  flag 2 := 0 ;
                  P(S) ;
                  V(P2) ;
                  end ;
             goto T  ;
 Ev 1 :      begin
             if événement 1 then
                  begin
                  flag 1 := 1 ;
                  V(S) ;
                  P(P1) ;
                  end ;
             else
             goto Ev 1 ;
             end
```

```
Ev 2  :     begin
            if événement 2 then
                  begin
                  flag 2 := 1 ;
                  V(S) ;
                  P(P2) ;
                  end
            else
            goto Ev 2 ;
            end ;
          parend ;
      end ;
```

On voit facilement qu'au moment où l'on entre pour la première fois dans T seules quatre configurations sont possibles :

- aucun événement n'est arrivé flag 1 et flag 2 valent 0 ; S vaut également 0;
- l'événement 1 est arrivé flag 1 = 1, flag 2 = 0, S = 1 ;
- l'événement 2 est arrivé flag 1 = 0, flag 2 = 1, S = 1 ;
- l'événement 1 et l'événement 2 sont arrivés flag 1 = 1, flag 2 = 1, S = 2.

Dans le premier cas, le processus T se bloquera et restera aussi longtemps bloqué qu'un des processus Ev 1 ou Ev 2 n'aura pas effectué V(S).

Dans le deuxième cas, le sémaphore S valant 1 l'opération P pourra être terminé ; au moment où l'on effectue l'opération V, s peut valoir soit 0, soit 1, si entre temps l'événement 2 est arrivé.

Prenons d'abord la première hypothèse : S vaut à nouveau 1 après l'opération V ; on remet à 0 flag 1 et l'on autorise une nouvelle arrivée de l'événement 1 ; l'opération P(S) peut être passée et le processus continue en testant le flag 2.

Dans la deuxième hypothèse ou si l'événement 2 est arrivé pendant que l'on traitait l'événement 1 ,flag 2 est à 1 et le sémaphore s a la valeur 1 ; la troisième opération P sera donc également passante et ,si l'événement 1 ne s'est pas produit à nouveau pendant le traitement de l'événement 2, s vaut 0 lorsqu'on retourne en Tr.

Le troisième cas s'examine de façon analogue.

Dans le quatrième cas, s a la valeur 2 lorsqu'on entre en Tr pour la première fois; on traite le premier événement, lorsqu'on effectue V(P1) s est égal à 1 ; il reste l'événement 2 à traiter.

Lorsqu'on revient en Tr, et si l'événement 1 n'est pas arrivé une nouvelle fois pendant le traitement de l'événement 2, s est égal à 0. On voit par conséquent que si flag 1 et flag 2 sont tous les deux à zéro lorsqu'on effectue l'instruction Tr,l'opération P qu'elle comprend ne pourra être terminée et l'on ne parcourt jamais la boucle sans qu'il y ait un travail à effectuer.

Remarquons que s'il n'en était pas ainsi,le processus Tr ne "s'arrêterait" jamais puisque la première opération P est suivie d'une opération V sur le même sémaphore .

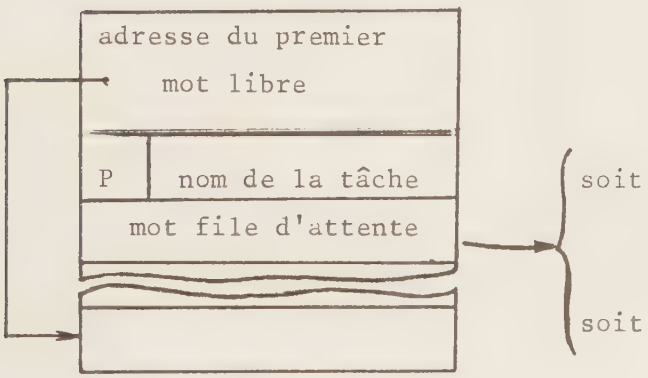
ANNEXE 2

Détails d'implémentation et mode d'emploi du régulateur.

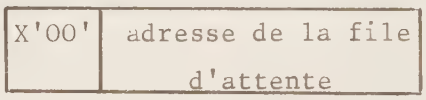
Les renseignements concernant les tâches gérées sont consignées dans 4 tables :

- table des tâches activées TACT

TACT



si la tâche n'a pas de file d'attente

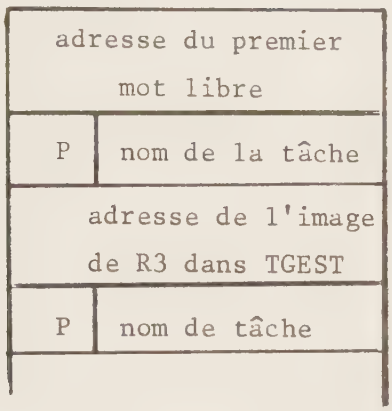


si une file d'attente est associée à la tâche.

deux mots sont nécessaires par tâche activée, les tâches sont rangées dans l'ordre d'activation.

- table des priorités TPRIOR

TPRIOR



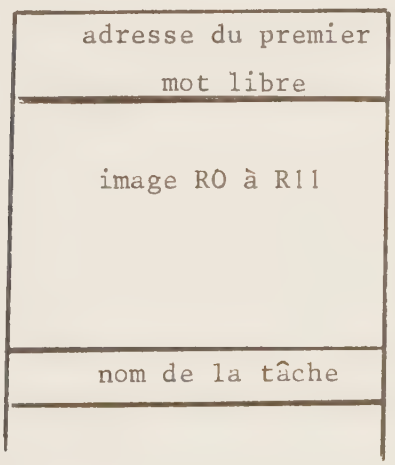
1 entrée

Les tâches sont rangées dans l'ordre des priorités décroissantes ; les tâches de même priorité sont rangées dans l'ordre d'activation ou de changement de la priorité.

- table de gestion TGEST

A chaque tâche on associe une zone de travail du régulateur

TGEST



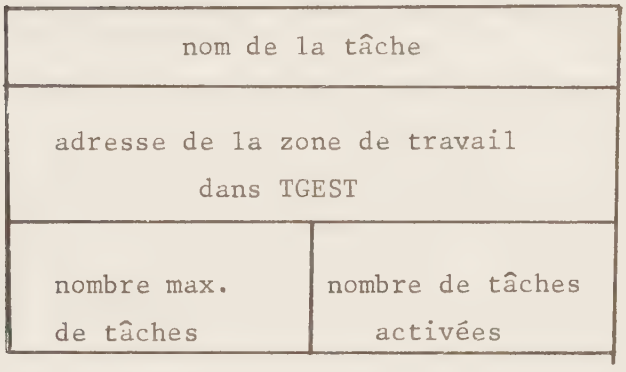
1 entrée

Cette zone est destinée à recevoir l'image des 12 premiers registres de la tâche au moment où celle-ci exécute BAL,8 ATMLT ; elle contient également le nom de la tâche.

Dans TPRIOR l'adresse de l'image du registre 3 permet de savoir quels sont les événements attendus par la tâche.

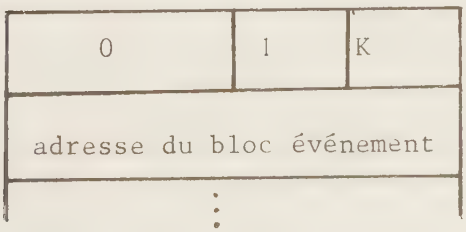
- table de la tâche lancée TCHEEX

TCHEEX



Par ailleurs, dans le mot ADRRET on stocke l'adresse de l'instruction qui suit l'appel à BEGP.
Elle constitue l'adresse de retour lorsqu'il n'y a plus aucune tâche activée.
Pour les besoins de la macro-instruction MLTW (attente d'événements multiples) on a créé TABLE :

TABLE



k est le nombre total d'événements attendus.

A chaque événement attendu correspond l'adresse du bloc événement associé.

Deux autres tables sont utilisées : TBADR1 et TBADR2 contenant respectivement les adresses dans TPRIOR des adresses de l'image de R3 dans TGEST et les adresses des éléments des tables d'événements attendus.

Sous-programme ACTIV - activer une tâche P - priorité $0 \leqslant P \leqslant 255$

appel R3 : | P | nom de la tâche | nom de la tâche : adresse de la première instruction

R4 : | X'80' | (lorsqu'il n'y a pas de file d'attente)

ou

| 0 | adresse de la file d'attente |

BAL,8 ACTIV

Seuls modifiés R3, R4 et R12 qui contient le code de retour :

R 12 à 0 tout s'est bien passé, tâche activée

premier octet à X'80' la tâche n'a pas été activée car le maximum de tâches est déjà atteint

- " - à X'01' tentative de double activation d'une tâche, la deuxième non prise en compte.

Le contrôle est dans tous les cas rendu à l'instruction qui suit BAL,8 ACTIV.

Sous-programme DSACTV - désactiver une tâche

appel R3 | nom tâche à désactiver |

BAL,8 DSACTV

Le seul registre modifié est R3 qui contient le code de retour.

R3 à 0 tout O.K.

1er octet de R3 à X'80' tâche inconnue ; rien n'a été changé.

Si la tâche se désactive elle-même, on relance la tâche la plus prioritaire prête.

S'il n'y a plus de tâches activées, on se branche à l'instruction qui suit BAL,8 BEGP.

Quand l'appel à DSACTV concerne une autre tâche que celle lancée, on continue en séquence dans le corps de la tâche.

BAL,8 DSACTV ne doit pas être programmé avant le lancement du régulateur.

sous-programme ASPRIOR - assigner une priorité à une tâche

appel

R3 | P | nom de la tâche

P est la nouvelle priorité de la tâche qui doit obligatoirement avoir été activée. Une tâche peut changer la priorité d'une autre tâche ou la sienne.

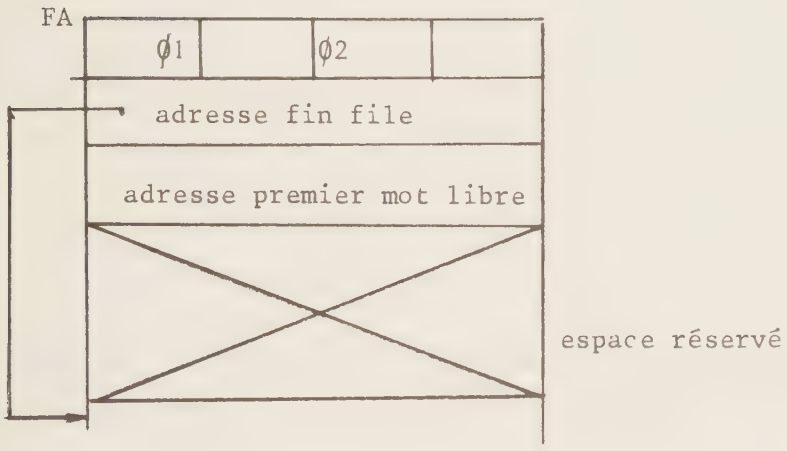
BAL,8 ASPRIOR

Aucun registre n'est modifié sauf R3 qui contient le code de retour.

R3 à 0 tout O.K.

1er octet de R3 à X'01' tâche inconnue ou pas de tâches activées.
Le contrôle est rendu dans tous les cas à l'instruction qui suit BAL,8 ASPRIOR.
La tâche est rangée dans la table des priorités TPRIOR derrière les tâches de même P, s'il y en a.

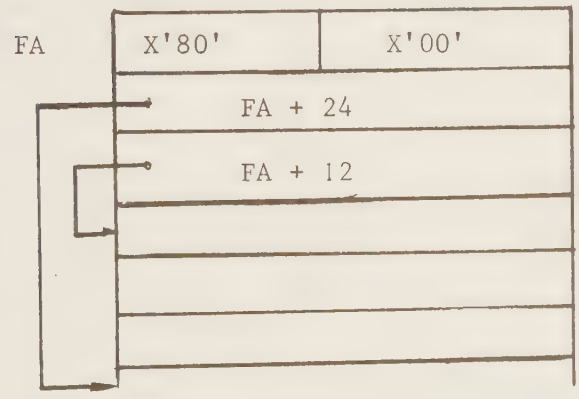
Sous-programme ENQ placer un mot dans la file d'attente de la tâche.
Chaque file d'attente doit être organisée de la façon suivante :



ϕ1 octet de contrôle 1 : X'80' file vide / X'00' non vide

ϕ2 octet de contrôle 2 : X'80' file pleine / X'00' non pleine

Voici l'exemple d'une file pouvant contenir 3 mots utiles :




```
FA      BOUND       4
        DATA,4,4    X'80000000'
        DATA,4,4    FA + 24
        DATA,4,4    FA + 12
        RES,4       3
```

Avant l'appel au sous-programme, les registres R3 et R4 doivent contenir respectivement le nom de la tâche dans la file d'attente de laquelle on veut placer l'information et le mot à introduire :

R3 : nom de la tâche

R4 : l'information à introduire

```
        BAL,8   ENQ
```

Aucun registre sauf R3 n'est modifié ; R3 contient le code de retour :

	R3 à 0	tout s'est bien passé
1er octet de R3	X'01'	tâche inconue
	X'02'	pas de file d'attente pour la tâche
	X'03'	file d'attente pleine, le mot ne peut être accepté
	X'80'	le mot a été placé et on vient de remplir la file.

Ce dernier cas n'est pas une erreur, il permet de savoir que la file vient d'être remplie, et ne pourra donc accepter de nouvelle entrée.

Sous-programme DEQ extraire le mot le plus ancien de la file d'attente.
pas de paramètres d'appel.

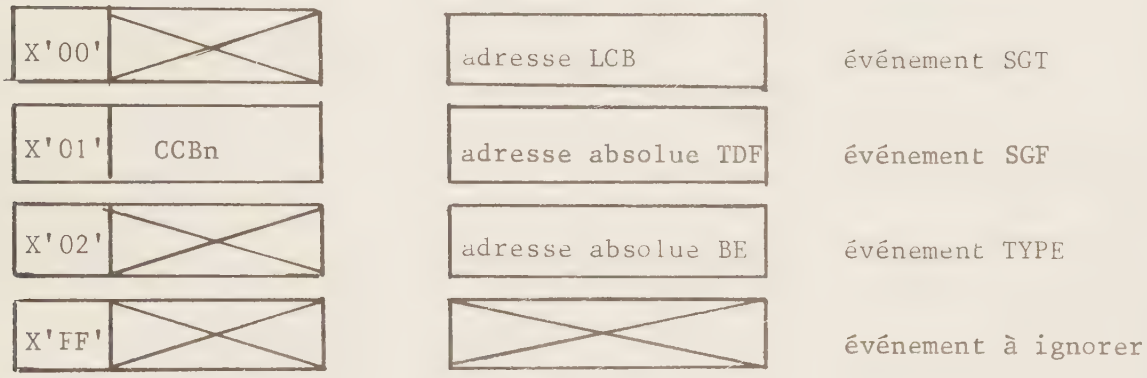
```
BAL,8     DEQ
```

le mot prélevé, si tout s'est bien passé, se trouve dans R4.
R3 et R4 sont les seuls registres modifiés.
R3 contient le code de retour :

	R3 à 0	tout O.K.
1er octet de R3	à X'01'	pas de file d'attente - erreur de programmation
	à X'02'	DEQ avant BEGP - erreur de programmation
	à X'80'	la file d'attente était vide, rien dans R4.

Sous-programme ATMLT - rendre le contrôle au régulateur et attendre l'arrivée d'un événement parmi n.

Il faut tout d'abord définir une table des événements attendus (TEA).



La table comporte un double mot par événement attendu :
Une entrée marquée X'FF' sera ignorée ;
adresse LCB est l'adresse de la table Line Control Bloc ;
CCBn - numéro du bloc de gestion directive (Command Control Bloc) correspondant à l'opération analysée ;
adresse TDF - adresse absolue de la table de description du fichier ;
adresse BE - adresse absolue du bloc événement associé à l'événement analysé.

Les paramètres d'appel de ATMLT doivent se trouver dans le registre 3 :

n est le nombre d'événements susceptibles de se produire décrits dans la TEA. n peut être égal à 0 ; l'adresse de la TEA n'est pas alors prise en compte, la tâche est considérée comme prête,

```
BAL,8      ATMLT
```

On relance la tâche la plus prioritaire prête ; la tâche retrouve ses registres, R0 à R 11 ; R 12 contient l'adresse absolue de l'élément de la TEA pointant vers l'événement arrivé.

Pour un événement SGF la détection d'une fin de fichier (ou début de fichier en lecture arrière) renvoie sur la séquence de fin de fichier déclarée par l'argument EOF de la description du fichier.

ANNEXE 3

Langage de commande :

Les commandes admises sont les suivantes :

1) * ABORT
2) * ESOURCE
3) * EPUITS
4) * ROMPRE
5) * FERMER

1) * ABORT ; pas de paramètres ; c'est une commande de mise au point ; elle sera supprimée par la suite.
Elle provoque l'abandon du travail puisqu'on exécute le procédure "abort" du moniteur.

2) * ESOURCE : établir sous-canal SOURCE

<mot clé> ::= DEST | EXP

<identifieur> ::= <idex fichier> | <device>

DEST est le destinataire, il est identifié par le code alphanumérique du composant :

code α-numérique	code numérique interne	signification
CR	01	lecteur de cartes
CP	02	perforateur de cartes
PR	05	imprimante
MT	06	dérouleur de bande
RD	07	disque amovible
FD	08	disque fixe
MD	0F	dimas

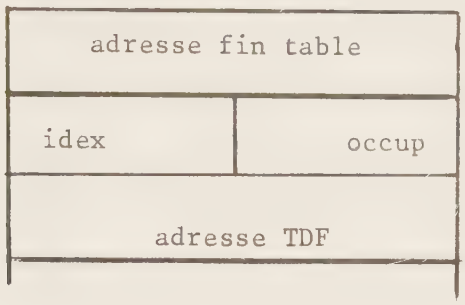
EXP est l'expéditeur ; il est identifié par l'idex du fichier.

Le traitement de la commande * ESOURCE implique

1. vérification existence et disponibilité fichier et composant
2. allocation mémoire pour la création des tampons composant.

Les renseignements concernant les fichiers sont consignés dans la table des fichiers connus (TFC)

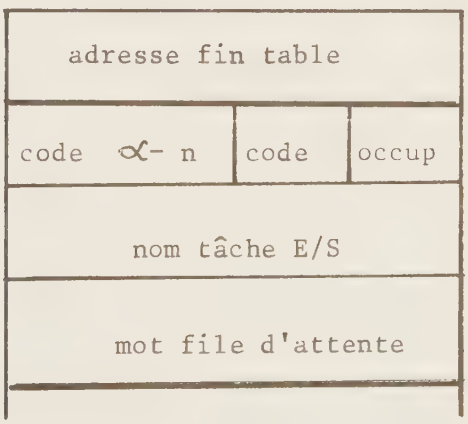
TFC



$$\text{occup.} = \begin{cases} 0, & \text{si fichier disponible} \\ \text{code}, & \text{si fichier occupé} \\ \text{dvt} & \end{cases}$$

TDF est la table de description du fichier créée par le Système.
Les composants admis (mais pas nécessairement connectés) sont décrits dans la table TCR :

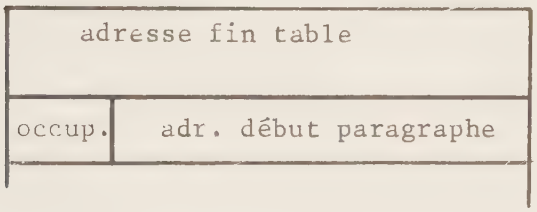
TCR



$$\text{occup.} = \begin{cases} 0, & \text{si device disponible} \\ \text{n° SC}, & \text{si occupé} \\ \text{X'80'}, & \text{si inexistant} \end{cases}$$

L'allocation du tampon composant est confiée au sous-programme ALM dans lequel on fait appel à la paragraphisation ; les tables TMTCS et TMTCP servent à la gestion des paragraphes. Elles contiennent 1 mot par paragraphe :

TMTC $\begin{matrix} S \\ P \end{matrix}$



occup = 0, si paragraphe libre
n° sous-canal, si alloué

3) * EPUITS : établir sous-canal PUITS
<mot clé> ::= DEST | EXP
<identifieur> ::= <idex fichier> | <device>
La commande n'est pas encore implémentée.

4) * ROMPRE : rompre sous-canal
la syntaxe est la même que celle de * EPUITS, la commande n'est pas encore implémentée.

5) * FERMER : "on ferme".
pas de paramètres pour cette commande.
Son traitement consiste à remettre à zéro le bloc événement FERM, ce qui interdira par la suite l'établissement de toute nouvelle liaison et provoquera la fin normale du programme, lorsque tous les échanges initiés avant la réception de la commande seront terminés.
La syntaxe des messages reçus à la machine à écrire est testée et l'utilisateur est averti des erreurs éventuelles par les messages suivants :

- COMMANDE ILLEGALE LIAISON UC-UC

  si les 9 premiers caractères introduits n'ont pas de sens pour le programme.

- COMMANDE ERRONEE ERREUR N - UC - UC

  si une erreur a été détectée dans une commande de nom valide, N précise alors la nature de l'erreur ;

  0 - syntaxe erronée
  1 - double définition du paramètre
  2 - mot clé erroné
  3 - nombre paramètres erroné
  4 - manque de paramètres.

  Si une commande valide ne peut pas être exécutée, on avertit l'usager :

- COMMANDE REFUSEE MOTIF : ---- ---- UC - UC

  idex
  code &- n
  composant

motif :

0 - nombre maximum sous-canaux dépassé ou fermeture demandée
1 - fichier inconnu
2 - fichier indisponible
3 - device occupé ou inconnu
4 - composant inexistant (non rattaché) (correspondant)
6 - composant indisponible (correspondant)
7 - manque place mémoire
8 - manque place mémoire (correspondant).

Sur réception de la commande * FERMER le message COMMANDE ENREGISTREE UC - UC apparaît sur la console.

BIBLIOGRAPHIE

[1] K. A. Bartlett : Transmission Control in a Local Data Network
in Information Processing 68 (p.p. 704 - 708),
North-Holland Publishing Company, Amsterdam 1969

[2] F. Behr : Protection contre les erreurs de transmission
in Colloque International sur la Téléinformatique (p.p. 714 - 723),
Chiron, Paris 1969

[3] D. Borel : Quelques résultats nouveaux sur l'emploi des codes correcteurs
in Colloque International sur la Téléinformatique (p.p. 724 - 733),
Chiron, Paris 1969

[4] D. W. Davies : The Principles of a data communication network for computers and remote peripherals
in Information Processing 68 (p.p. 709 - 715),
North-Holland Publishing Company, Amsterdam 1969

[5] Davies : A versatile Data communication Network
in Colloque International pour la Téléinformatique (p.p. 461 - 467),
Chiron , Paris 1969

[6] E. W. Dijkstra : Cooperating Sequential Processes
Technological University, Eindhoven 1965

[7] E. Gelenbe : The two-thirds rule for dynamic storage allocation under équilibrium
in Information Processing Letters Vol 1 n°2 (p.p. 59 - 60)
North-Holland Publishing Company, Amsterdam 1971

[8] E. Gelenbe : Communication personnelle

[9] M. Grandidier et alteri : Présentation d'un système de transmission des informations numériques permettant d'effectuer l'entrée des travaux à distance.
in Colloque International pour la Téléinformatique (p.p. 562 - 570),
Chiron, Paris 1969

[10] A. J. Henry : Les satellites de télécommunications et de téléinformatique
in Colloque International pour la Téléinformatique (p.p. 74 83),
Chiron, Paris 1969

[11] George M. Hicken : Information Networks of Computers
in Fourth Generation Computers : user requirements and transition (p.p. 31 - 58)
1970

[12] Joint ICL/CII/CDC Communication Control Procedure Standards
GOF 0/12/3 February 1971
Terminal/communication Sub-System Architecture

[13] R. Kierbel e. a. : Exemple de liaison à 50 kbauds entre calculateurs du CNET
in Colloque International sur la Téléinformatique (p.p. 87 - 91),
Chiron, Paris 1969

[14] James Martin : Telecommunications and the computer
Prentice-Hall,
Englewood Cliffs, N. J. 1969

[15] M. R. de Rocquigny: Expérimentation d'une liaison à haute vitesse et ses perspectives
in Colloque International sur Téléinformatique
(p.p. 1079 - 1081)
Chiron, Paris 1969

[16] R. A. Scantlebury e. a. : The design of a message switching center for a digital communication network
in Information Processing 68 (p.p. 723 - 727),
North-Holland Publishing Company, Amsterdam 1969

[17] Schramel : DS 714 A message switching system for telegraph and data communication
in Colloque International sur la Téléinformatique (p.p. 383 - 392),
Chiron, Paris 1969

[18] Jeffrey D. Stone : The large Scale Data Processing Network
in Colloque International sur la Téléinformatique
(p.p. 478 - 486),
Chiron, Paris 1969

ACADÉMIE DE PARIS

UNIVERSITÉ DE PARIS VI

4, Place Jussieu - 75230 PARIS-CEDEX 05

SERVICE DE LA SCOLARITE

Bureau des Enseignements du 3e cycle

TÉLÉPHONE : 336-25-25 - 325-12-21

PERMIS D'IMPRIMER

DOCTORAT : DOCTEUR INGENIEUR

THESE DE :

NOM : Monsieur BRANDWAJN

Prénoms : Alexander

né le 13 avril 19 48 à HOHNE CAMP Dépt : ALLEMAGNE

1ère THESE : 3 "Procédure d'exploitation pour une liaison uc-uc"

2e THESE (Propositions faites par l'Université) :

PRESIDENT : M. ARSAC

EXAMINATEURS : M. DUPUY
M. FEAUTRIER

Vu et Approuvé

PARIS, le 9 octobre 1972.

Le Président

de l'Université de PARIS VI

Sc. 3e c. 72-09